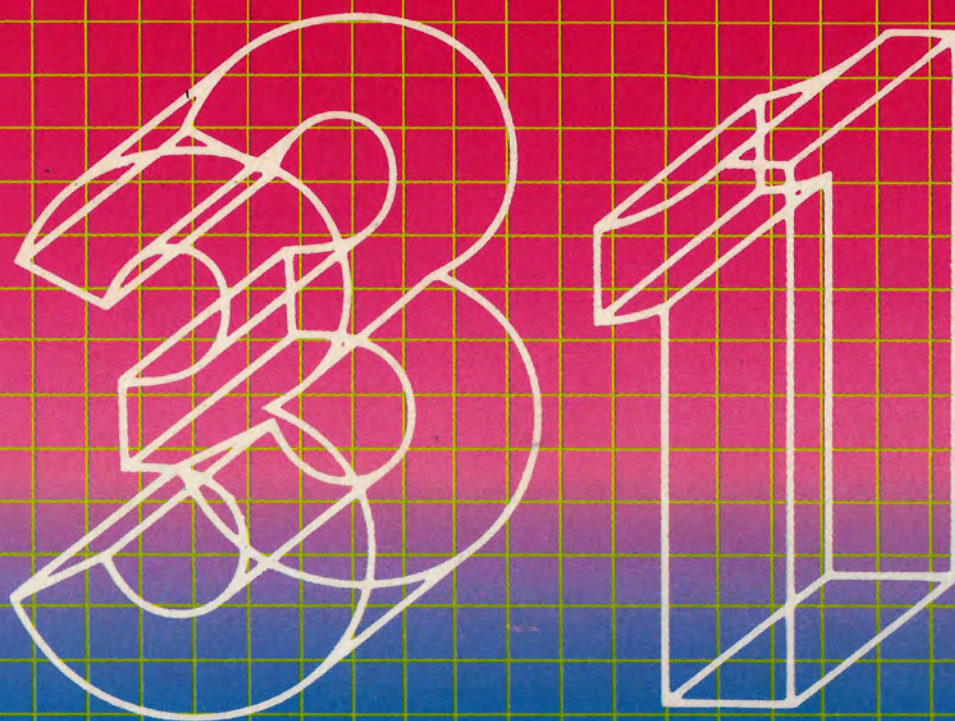


BIBLIOTECA PRACTICA

TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD,
IBM, SPECTRUM, COMMODORE Y MSX



¿QUIERE APRENDER INFORMATICA?

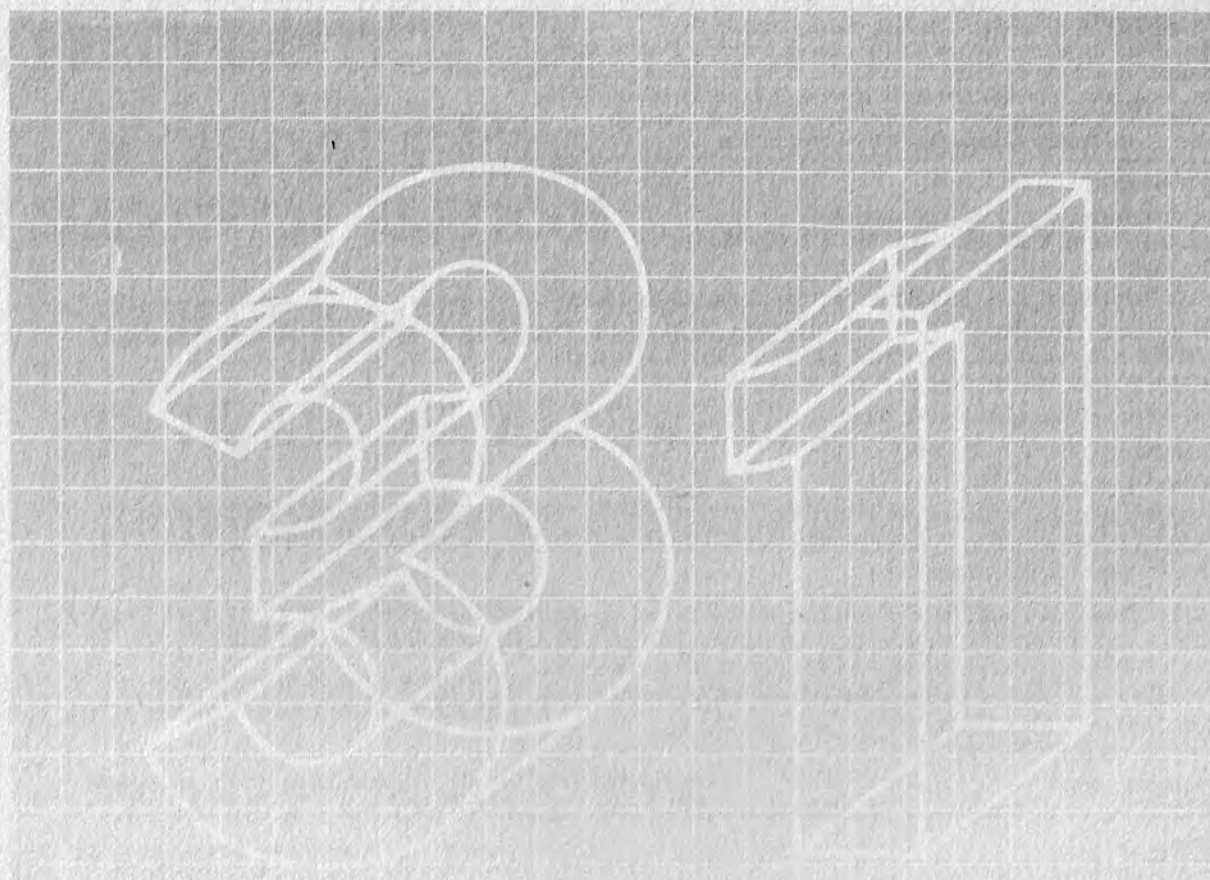
PRACTIQUE CON SU ORDENADOR

EDICIONES SIGLO CULTURAL

BIBLIOTECA PRACTICA TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD,
IBM, SPECTRUM, COMMODORE Y MSX



EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S. A.

Director-editor:

RICARDO ESPAÑOL CRESPO.

Gerente:

ANTONIO G. CUERPO.

Directora de producción:

MARIA LUISA SUAREZ PEREZ.

Director de la colección:

JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño:

BRAVO-LOFISH.

Maquetación:

D. S. M.

Dibujos:

JOSE OCHOA

Tomo 31. «Experiencias y prácticas en logo», Carlos Albert, Técnico de Informática; Adoración Llena Jubero, Técnico de Informática. «Manejo de sprites y elementos gráficos», «Trucos y rutinas básicas», Francisco Morales. Técnico de Informática. «Aprende con el ordenador», AULA DE INFORMATICA APLICADA: Jesús Bocho, Licenciado en Informática; Alejandro Marcos, Licenciado en Química, Francisco Javier Gavilán, Diplomado de Telecomunicación, «El Taller de Hardware», Carlos Rey, Ingeniero Industrial. «Pequeña historia de la Informática», «Temas monográficos de vanguardia», «Terminología», «Vocabulario de Informática», Blanca Arbizu, Técnico de Informática.

Ediciones Siglo Cultural, S. A.

Dirección, redacción y administración:

Pedro Teixeira, 8-2.º. Teléf. 455 09 99. 28020 Madrid.

Publicidad:

Gofar Publicidad, S. A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:

COEDIS, S. A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.

Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:

CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.

Buenos Aires - 1.290. Argentina.

Todos los derechos reservados. Este libro no puede ser, en parte o totalmente, reproducido, memorizado en sistemas de archivo, o transmitido en cualquier forma o medio, electrónico, mecánico, fotocopia o cualquier otro, sin la previa autorización del editor.

ISBN del tomo: 84-7688-153-3

ISBN de la obra: 84-7688-047-2

Fotocomposición:

ARTECOMP, S. A. Albarracín, 50. 28037 Madrid.

Imprime:

GRÁFICAS EMA. (Madrid)

© Ediciones Siglo Cultural, S. A., 1986

Depósito legal: M-43.185-1986

Printed in Spain - Impreso en España.

Suscripciones y números atrasados:

Ediciones Siglo Cultural, S. A.

Pedro Teixeira, 8-2.º. Teléf. 259 73 31. 28020 Madrid.

Agosto, 1987

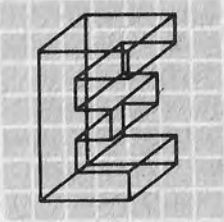
P.V.P. Canarias: 355.—Ptas.

INDICE

■ EXPERIENCIA Y PRACTICAS EN LOGO	5
■ MANEJO DE SPRITES Y ELEMENTOS GRAFICOS	14
■ TRUCOS Y RUTINAS BASICAS	24
■ EL TALLER DEL HARDWARE	33
■ APRENDER CON EL ORDENADOR	39
■ PEQUEÑA HISTORIA DE LA INFORMATICA	46
■ TEMAS MONOGRAFICOS DE VANGUARDIA	49
■ TERMINOLOGIA	52
■ VOCABULARIO DE INFORMATICA	55

EXPERIENCIA Y PRACTICAS EN LOGO

Lista de propiedades



N Logo a las variables y a los procedimientos podemos asociarles una lista de propiedades.

¿Qué es una lista de propiedades?

La estructura es la siguiente:

[propiedad 1 valor 1 propiedad 2 valor 2 ...]

Las listas de propiedades contienen pares de elementos formados por una propiedad y su valor.

Una lista de propiedades con un solo elemento es:

[LIBRO PEQUEÑO]
propiedad 1 — LIBRO
valor 1 — PEQUEÑO

Una con dos pares de elementos

es:

[PADRE JOSE MADRE MARIA]
propiedad 1 — PADRE
valor 1 — JOSE
propiedad 2 — MADRE
valor 2 — MARIA

No es posible asociar una lista de propiedades a una variable que no tenga ningún valor.

NOTA: El Logo del SPECTRUM no dispone de la posibilidad de asociar listas de propiedades a las variables y/o a los procedimientos.

Las listas de propiedades son gestionadas por diversas primitivas, veamos alguna de ellas.

PONPROP nombre propiedad objeto

nombre: es el nombre de la variable o del procedimiento al que vamos a asociar una propiedad.

propiedad: es el nombre de la propiedad a asociar. Si esta propiedad no está en la lista de propiedades del nombre especificado se incluirá en ella en el primer lugar.

objeto: es el valor que toma la propiedad especificada.

? HAZ "A "CASA

? PONPROP :A "TELEFONO "ROJO

Asociamos a la variable A la propiedad "TELEFONO con el valor "ROJO.

Si a la misma variable queremos darle más propiedades, hacemos lo mismo.

? PONPROP :A "BALCON "EXTERIOR

Para comprobar que la variable A tiene asociadas estas propiedades con sus respectivos valores, utilizamos la primitiva:

? PROP nombre propiedad

Esta primitiva devuelve el valor de la propiedad especificada de la lista asociada al nombre indicado:

nombre: es el nombre de la variable o del procedimiento del que vamos a consultar la lista de propiedades.

Una variable o procedimiento pueden tener asociada una lista de propiedades.

EXPERIENCIA Y PRACTICAS EN LOGO

propiedad: es el nombre de la propiedad que queremos consultar.

Si la propiedad que especificamos no tiene valor, la primitiva PROP devuelve la lista vacía.

Si deseamos saber los valores de las propiedades asociadas a la variable A en los ejemplos anteriores, teclearemos:

```
? ES PROP :A "TELEFONO  
ROJO
```

ROJO es el valor de la propiedad TELEFONO.

```
? ES PROP :A "BALCON  
EXTERIOR
```

```
? ES PROP :A "TERRAZA
```

En la pantalla no aparece nada, pues la variable A no tiene asociada ninguna propiedad con el nombre TERRAZA.

Para cambiar el valor a una propiedad determinada, por ejemplo, a BALCON, teclearemos:

```
? PONPROP :A "BALCON "INTERIOR
```

Comprobemos que el valor actual de la propiedad BALCON es INTERIOR.

```
? PROP :A "BALCON  
INTERIOR
```

Si borramos del espacio de memoria una variable que tiene una lista de propiedades asociadas, las perdemos.

Un motivo de error es consultar la lista de propiedades de una variable que no tiene valor. Logo muestra el mensaje:

nombre NO TIENE VALOR

Sigamos con los ejemplos.

```
? PARA "A1  
> HAZ "A [ ME ENCANTAN LAS ZAHORIAS ]  
> ES :A  
> FIN
```

Ahora le asignamos al procedimiento A1 las propiedades que deseamos.

```
? PONPROP "A1 "COMIDAS "BUENAS  
? PONPROP "A1 "ALIMENTOS "NATURALES
```

```
? ES PROP "A1 "COMIDAS  
BUENAS
```

```
? ES PROP "A1 "ALIMENTOS  
NATURALES
```

El valor de una propiedad puede ser cualquier objeto Logo, veamos cuando el objeto es una lista.

```
? PARA A1  
> HAZ "A "FAMILIA  
> PONPROP :A "PADRE [ JOSE ANTONIO ]  
> PONRPOP :A "MADRE [ MARIA LUISA ]  
> ES PROP :A "PADRE  
> ES PROP :A "MADRE  
> FIN
```

Si ejecutamos este procedimiento:

```
? A1
```

En la pantalla aparece:

```
JOSE ANTONIO  
MARIA LUISA
```

La orden DEVUELVE

Ya vimos que el LOGO disponía de una orden con la que podíamos detener o interrumpir un procedimiento que se estaba ejecutando, devolviendo el control, bien al procedimiento que lo había llamado, bien al nivel superior (al usuario).

Esta orden es:

ALTO

Otra similar de que dispone el LOGO y cuya función es similar a ALTO es:

DEVUELVE objeto

y en su forma abreviada:

DEV objeto

Esta primitiva detiene la ejecución del procedimiento en que se encuentra y devuelve a su vez el objeto que se especifica.

Si definimos el siguiente procedimiento:

```
? PARA EJEMPLO  
> HAZ "C :A * :B
```

Una lista de propiedades está constituida por un número par de elementos.

```
> SI :C > 10 [DEV :C ]
> HAZ "A :A + 1
> HAZ "B :B + 2
> EJEMPLO
> FIN
```

e inicializamos las variables:

```
? HAZ "A 0
? HAZ "B 0
```

y ahora introducimos:

```
? ESCRIBE EJEMPLO
```

en la pantalla obtendremos:

18

El procedimiento se ejecuta hasta que se interrumpe cuando la condición establecida se cumple y ejecuta la orden DEV.

El número que se obtiene corresponde al valor que tiene la variable C, que es el que se ordena que se devuelva en el procedimiento EJEMPLO. Por eso, al ordenar que se escriba EJEMPLO, lo que se escribe es el dato que devuelve el procedimiento.

Si introducimos:

```
? ESCRIBE 5 + EJEMPLO
```

en la pantalla obtenemos:

23

Este número corresponde al valor que devuelve el procedimiento EJEMPLO sumado a cinco.

Otro ejemplo que nos puede ayudar a comprender esta orden es el siguiente:

Definimos los siguientes procedimientos:

```
? PARA A
> DEV "CONTRO
> FIN

? PARA B
> DEV "LADOR
> FIN
```

Si ahora introducimos:

```
ESCRIBE PALABRA AB
```

en la pantalla obtenemos:

CONTROLADOR

Ordenamos que se escriba, formando una palabra, dos objetos. Pero en este caso los dos objetos son nombres de procedimientos.

Cada uno de estos procedimientos devuelve una palabra determinada, que son los datos que utiliza la orden:

```
? ESCRIBA PALABRA A B
```

El resultado sería el mismo si hubiésemos introducido:

```
? ESCRIBA PALABRA "CONTRO "LADOR
```

Algo parecido podemos realizar con las listas.

Si definimos los siguientes procedimientos:

```
? PARA C
> DEV [ EN UN LUGAR ]
> FIN

? PARA D
> DEV [ DE LA MANCHA ]
> FIN
```

y ahora introducimos:

```
? ES FRASE CD
```

en la pantalla aparecerá:

EN UN LUGAR DE LA MANCHA

La frase se forma con los datos que devuelven los procedimientos que se especifican (el procedimiento C devuelve EN UN LUGAR, y el procedimiento D, DE LA MANCHA).

Aquí tienes otro ejemplo curioso.

Definimos el procedimiento CALOR

```
? PARA CALOR
> DEV [ MUCHO FRIO ]
> FIN
```

Si ahora introducimos:

```
? ESCRIBE FRASE "HACE CALOR
```

en la pantalla aparece la siguiente frase:

HACE MUCHO FRIO

La frase se forma con la palabra HACE y con la lista que devuelve el procedimiento CALOR, MUCHO FRIO.

No es posible asociar propiedades a una variable que no tenga valor.

EXPERIENCIA Y PRACTICAS EN LOGO

Cuadro resumen

— PONPROP nombre propiedad objeto

Asocia a nombre la propiedad con el valor objeto especificado.

— PROP nombre propiedad

Esta primitiva devuelve el valor de la propiedad especificada de la lista asociada al nombre indicado.

— DEVUELVE objeto

Interrumpe la ejecución del procedimiento en que se halla y devuelve como resultado el objeto que se especifica.

Ejercicios

1. *Define los procedimientos necesarios para realizar el siguiente juego:*

— El jugador deberá escoger una letra cualquiera, la que prefiera.

— Una vez introducida la letra deberá introducir palabras que empiecen por dicha letra, teniendo en cuenta:

Se concederá un tiempo de espera para introducir cada palabra.

Se podrán introducir hasta 10 palabras.

Si la palabra introducida no empieza por la letra elegida o si tras la espera no se ha tecleado nada, en la pantalla aparecerá el mensaje TIEMPO.

— Al final del juego se informará de los puntos obtenidos, y se ofrecerá la oportunidad de volver a jugar.

2. *Dibuja una calculadora.*

3. *Intenta realizar el siguiente dibujo:*

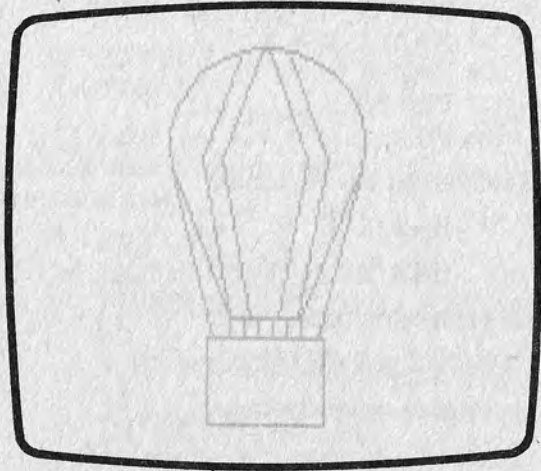


Fig. 1.

4. *Define los procedimientos necesarios para clasificar en dos columnas diferentes los números pares e impares que quieras introducir.*

5. *¿Son correctas las siguientes órdenes?:*

- ? PARA "A1
 - > HAZ "A "CENICERO
 - > PONPROP :A "GRANDE "RICO
 - > ES PROP :A "GRANDE
 - > FIN
- ? PARA A2
 - > HAZ "A "JARDIN
 - > HAZ "B PONPROP :A "FLOR "AMARILLA
 - > FIN
- ? PARA A3
 - > PONPROP "A3 "MUSICA "POP
 - > ES PROP "A3 "MUSICA
 - > FIN
- ? PARA A4
 - > PARA A4
 - > HAZ "A 0
 - > DEV :A
 - > FIN
- ? PARA A5
 - > DEV "PRONTO
 - > FIN

La primitiva PROP nos permite consultar el valor de una propiedad.

? ES PALABRA "HASTA A5
 ? DEV [NO ES POSIBLE]

■ Solución a los ejercicios

1:

Definimos el procedimiento J para:

inicializar la pantalla
 leer la letra elegida
 ejecutar el procedimiento JUEGO (parte central del juego)
 ofrecer la oportunidad de volver a jugar

```
? PARA J
> PT
> BT
> PONCURSOR [ 5 1 ]
> ESCRIBE [ ELIGE UNA LETRA... ]
> PONCURSOR [ 25 1 ]
> HAZ "T LC
> PONCURSOR [ 25 1 ]
> ES :T
> JUEGO 5 :T 0
> PONCURSOR [ 2 22 ]
> ES [ ¿QUIERES JUGAR OTRA VEZ?
  ( S / N ) ]
> PONCURSOR [ 37 22 ]
> HAZ "S LC
> SI :S = "S [ ] [ BT ALTO ]
> FIN
```

El procedimiento JUEGO tiene tres entradas:

X posición del cursor
 T letra elegida
 P total puntos

Este procedimiento se ejecutará 10 veces, desde que x = 5 hasta que x = 14.

```
? PARA JUEGO :X :T :P
> PONCURSOR LISTA 1 :X
> ESPERA 20
> SI TECLA? [ HAZ "P CORREC :T :P ]
  [ ES "TIEMPO ]
> SI :X = 14 [ PONCURSOR [ 15 7 ] ES
  (FRASE [ TOTAL PUNTOS =) :P )
  ALTO ] [ JUEGO (:X + 1) :T:P ]
> FIN
```

El jugador debe tener un poco de tiempo para pensar, por tanto, establecemos una

espera (ESPERA 20); si tras la espera se detecta que se han introducido caracteres, se ejecuta el procedimiento CORREC, si no, en la pantalla aparece el mensaje TIEMPO.

El procedimiento CORREC tiene dos entradas

```
T letra elegida
P total puntos
? PARA CORREC :T :P
> HAZ "A LP
> SI VACIO? :A [ PONCURSOR LISTA
  1 (-1 + ULTIMO CURSOR) ES "TIEMPO ]
  [ HAZ "C PRIMERO :A SI :C = :T ]
  [ HAZ "P :P + 1 ] ]
> DEVUELVE :P
> FIN
```

Este procedimiento se encarga de:

- leer la palabra;
- controlar que no sea la palabra vacía, en cuyo caso se considera error;
- comprobar si la primera letra de la palabra es igual a la letra elegida;
- incrementar en 1 el contador de puntos cuando proceda;
- y
- devolver al procedimiento JUEGO el valor de marcador.

Este valor, que devuelve el procedimiento CORREC al procedimiento JUEGO a través de la primitiva DEVUELVE, es almacenado en la variable P.

Empecemos a jugar

? J

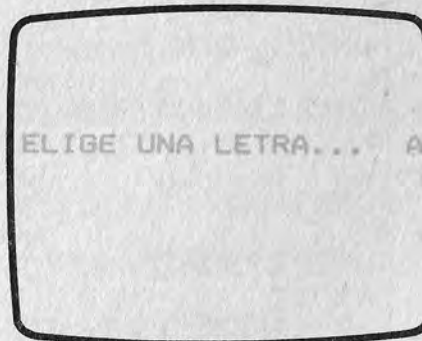


Fig. 2.

La orden DEVUELVE interrumpe la ejecución del procedimiento en que está.

EXPERIENCIA Y PRACTICAS EN LOGO

(Pulsamos la tecla A)
y todas las palabras que podamos que empiecen por A.

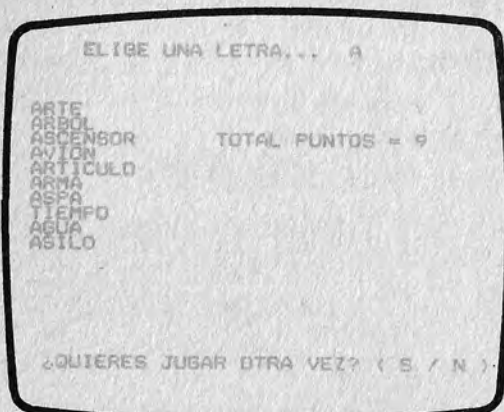


Fig. 3.

No está mal, hemos conseguido 9 puntos.

Si deseas volver a jugar, contesta a la pregunta afirmativamente (pulsas la tecla S).

2:

Dibujamos una calculadora con este procedimiento:

```
? PARA C
> BP
> SL
> PONCL 3
> PONPOS [ -80 44 ]
> PONRUMBO 90
> BL
> REPITE 2 [ AV 160 GD 90 AV 80 GD 90 ]
> SL
> PONPOS [ -70 39 ]
> BL
> REPITE 2 [ AV 65 GD 90 AV 15 GD 90 ]
> SL
> PONX 5
> BL
> REPITE 4 [ REPITE 4 [ AV 15 GD 90 ] AV 15 ]
> HAZ "A FRASE [ 7 8 9 / MRC ]
> HAZ "B FRASE [ 4 5 6 * M - ]
> HAZ "C FRASE [ 1 2 3 - M + ]
```

```
> HAZ "D FRASE [ 0 . + = ]
> PONCURSOR [ 18 12 ]
> ESCRIBE :A
> PONCURSOR [ 18 13 ]
> ESCRIBE :B
> PONCURSOR [ 18 14 ]
> ESCRIBE :C
> PONCURSOR [ 20 15 ]
> ESCRIBE :D
> HAZ "A "ON
> PONCURSOR [ 11 13 ]
> ESCRIBE [ 11 13 ]
> HAZ "A "OFF
> PONCURSOR [ 11 14 ]
> ESCRIBE :A
> PONCURSOR [ 16 9 ]
> ESCRIBE "523
> FIN
```

Ejecuta el procedimiento tecleando:

? C

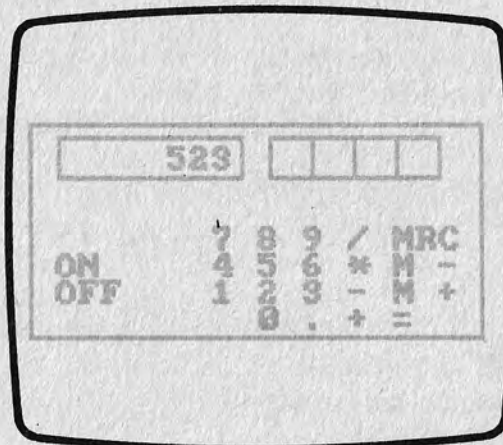


Fig. 4.

3:

Para realizar el dibujo del globo vamos a ir definiendo procedimientos que dibujan diferentes partes de éste.

En primer lugar, definimos el que dibuja lo que es el globo en sí:

```
? PARA A
> SL
> PONPOS [ -10 10 ]
> BL
> GI 20 AV 60
> GD 10
```

DEVUELVE sólo se puede introducir dentro de un procedimiento.

```

> REPITE 21 [ AV 6 GD 10 ]
> AV 60
> PONRUMBO 270
> AV 24
> FIN

```

El procedimiento B es el que dibuja, junto al C y el D, las franjas que tiene el globo:

```

? PARA B
> SL
> PONPOS [ -10 10 ]
> C 40
> SL
> PONPOS [ -3 10 ]
> C 43
> SL
> PONPOS [ 14 10 ]
> D 40
> SL
> PONPOS >D43 [7 10]
> FIN

```

C dibuja las franjas de la derecha:

```

? PARA C :V
> PONRUMBO 0
> BL
> GI 10 AV 60
> GD 30 AV :V
> FIN

```

D dibuja las franjas de la izquierda:

```

? PARA D :V
> PONRUMBO 0
> BL
> GD 10 AV 60
> GI 30 AV :V
> FIN

```

El procedimiento E es el que va a dibujar la cuerda que une el globo con la cesta:

```

? PARA E
> SL
> PONPOS [ -10 10 ]
> PONRUMBO 0
> REPITE 6 [ BL RE 7 AV 7 GD 90 SL
  AV 5 GI 90 ]
> FIN

```

F dibuja la cesta:

```

? PARA F
> SL
> PONPOS [ -18 -30 ]
> PONRUMBO 0

```

```

> BL
> REPITE 2 [ AV 32 GD 90 AV 41 GD
  90 ]
> AV 32 GD 90
> AV 2 GI 90
> GI 13 AV 65
> SL
> PONPOS [ 21 2 ]
> GD 26
> BL
> AV 65
> FIN

```

Una vez que tenemos todos los procedimientos necesarios definidos, los englobamos en GLOBO para realizar el dibujo completo e inicializar el proceso:

```

? PARA GLOBO
> BP
> OT
> A
> B
> E
> F
> FIN

```

Para ver el dibujo tenemos que introducir:

```

? GLOBO

```

4:

En primer lugar, definimos el procedimiento que va a dibujar la tabla donde se van a ir escribiendo los números que se introduzcan:

```

? PARA A
> SL
> PONPOS [ -105 -50 ]
> BL
> REPITE 2 [ AV 140 GD 90 AV 210 GD
  90 ]
> SL
> PONPOS [ -8 -50 ]
> BL
> AV 140 RE 16
> GD 90 RE 97
> AV 210
> PONCURSOR [ 8 4 ]
> ES [ NUM. PARES ]
> PONCURSOR [ 20 4 ]

```

Es posible cambiar el valor de una propiedad.

EXPERIENCIA Y PRACTICAS EN LOGO

```
> ES [ NUM. IMPARES ]
> FIN
```

El procedimiento B es el que solicita un número:

```
? PARA B
> BT
> PONCURSOR [ 2 21 ]
> ES [ INTRODUCE EL NUMERO QUE QUIERAS ]
> HAZ "A LL
> SI :A = [ ] [ B ]
> HAZ "A UL :A
> C
> B
> FIN
```

Una vez que el número se ha introducido, se pregunta si la lista es vacía, lo que significa que se ha pulsado RETURN sin introducir ningún valor. De ser así, vuelve a ejecutarse el procedimiento.

Si se ha introducido un número, se ejecuta el procedimiento C:

```
? PARA C
> HAZ "B RESTO :A 2
> SI :B = 0 [PONCURSOR LISTA
  12 :F ES :A ] [ PONCURSOR LISTA
  24 :F ES :A ]
> SI :F = 16 [ OTRA ] [ HAZ "F :F + 1 ]
> FIN
```

En este procedimiento se asigna a la variable B el resto de la división del número introducido (variable :A) entre dos.

Si el resto es igual a cero, significa que el número es par, por lo que situamos el cursor en el lugar correspondiente a los números pares y lo escribimos. Si el número es impar, lo escribiremos en la columna correspondiente a los números impares.

La variable F es la que controla las líneas en donde se van escribiendo los números. Si la F tiene un valor igual a 16, significa que la tabla está llena y se ejecuta el procedimiento OTRA. Si no es igual a 16, se incrementa en uno el valor de la variable y se devuelve el control al procedimiento B.

El procedimiento OTRA solicita que se pulse una tecla. Una vez pulsada se vuelve a comenzar el proceso:

```
? PARA OTRA
> BT
> PONCURSOR [ 12 22 ]
> ES [ PULSA UNA TECLA ]
> HAZ "T LC
> TODO
> FIN
```

Por último, definimos el procedimiento que inicializa el proceso y que lo pone en funcionamiento:

```
? PARA TODO
> BP
> BT
> OT
> A
> HAZ "F
> B
> FIN
```

Para ejecutarlo introducimos:

```
? TODO
```

La primera pantalla que obtenemos es:



Fig. 5.

Si empezamos a introducir números hasta completar la tabla, obtenemos la siguiente pantalla:

La primitiva DEVUELVE necesita que se le especifique el objeto Logo que se desea obtener.

NUM. PARES	NUM. IMPARES
4	?
8	
12	
54	
4	245
	43
	9
66	19

PULSA UNA TECLA

Fig. 6.

5:

```

- ? PARA "A1
> HAZ "A "CENICERO
> PONPROP :A "GRANDE "RICO
> ES PROP :A "GRANDE
> FIN

```

CORRECTO

Si ejecutamos el procedimiento A1, en la pantalla aparecerá la palabra RICO.

```

- ? PARA A2
> HAZ "A "JARDIN
> HAZ "B PONPROP :A "FLOR
  "AMARILLA

```

> FIN

INCORRECTO.

El Logo devuelve el mensaje:

PONPROP NO HA DEVUELTO NADA A

HAZ

```

- ? PARA A3

```

```

> PONPROP "A3 "MUSICA "POP
> ES PROP "A3 "MUSICA
> FIN

```

CORRECTO.

El procedimiento A3 devuelve la palabra POP.

```

- ? PARA A4
> HAZ "A 0
> DEV :A
> FIN

```

El procedimiento en sí es correcto. Dependería de cómo se ejecutase el que diese o no error.

Si ejecutamos el procedimiento:

```

? A4

```

obtendríamos el mensaje:

NO SE QUE DEBO HACER CON 0

Si introducimos:

```

? ES A4

```

obtendremos:

0

```

- ? PARA A5
> DEV "PRONTO
> FIN
? ES PALABRA "HASTA A5

```

CORRECTO.

En la pantalla aparecerá la palabra:

HASTAPRONTO

```

- ? DEV [ NO ES POSIBLE ]

```

INCORRECTO.

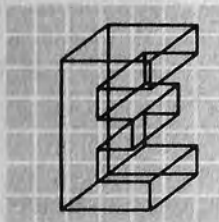
La orden DEVUELVE tiene que estar a la fuerza dentro de un procedimiento.

Los datos que se obtienen con la orden DEVUELVE se pueden utilizar en otros procesos.

Podemos formar palabras o frases con los datos que proporciona la primitiva DEVUELVE.

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

La alta resolución en el IBM PC y compatibles (II)



N el tomo anterior vimos algunos de los comandos gráficos que podíamos usar si trabajábamos en un IBM pc o compatible bajo GWBASIC. En este nuevo tomo veremos, por fin, cómo definir gráficos en dicho ordenador

La instrucción GET

Esta instrucción será la que nos permitirá, por así decirlo, definir nuestros propios gráficos. El tamaño de los mismos y el número de ellos sólo vienen dados por la cantidad de memoria libre que tengamos.

Con esta instrucción podemos almacenar en una variable numérica (entera o de simple precisión) el contenido de un área rectangular de la pantalla. Para ello tenemos que decirle al ordenador cuál es la esquina superior izquierda y la esquina inferior derecha del rectángulo perteneciente a la pantalla que queremos almacenar.

La sintaxis de esta instrucción es la siguiente:

GET (X1, Y1) - (X2, Y2), var

donde:

X1 = Coordenada en X de la esquina superior izquierda del rectángulo que queremos almacenar.

Y1 = Coordenada en Y de la esquina superior izquierda del rectángulo que queremos almacenar.

Y2 = Coordenada en Y de la esquina inferior derecha del rectángulo que queremos almacenar.

X2 = Coordenada en X de la esquina inferior derecha del rectángulo que queremos almacenar.

var = Variable numérica (a ser posible entera, para ahorrar memoria, donde quedará almacenado el rectángulo con el gráfico).

En la figura 1 podemos ver cómo están representadas las coordenadas del rectángulo que queremos almacenar con la sentencia GET.

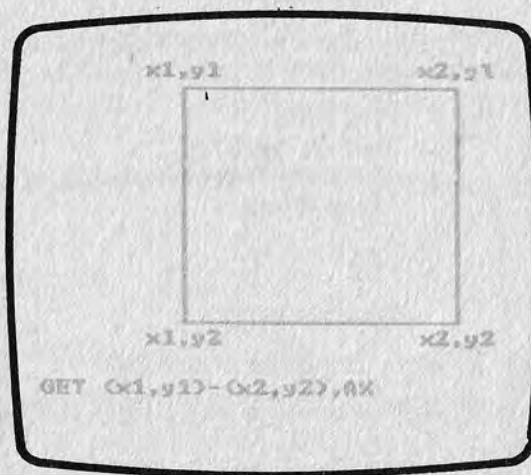


Fig. 1. Organización de las coordenadas de la instrucción GET.

La variable que se encargará de almacenar el gráfico tiene que haber sido dimensionada con anterioridad a la sentencia GET. Ha de ser un vector. Para saber el número de elementos que tiene dicho vector es necesario aplicar la siguiente fórmula:

$$\text{Número de elementos} = \text{INT} ((X2 - X1) / 4) * (Y2 - Y1 - 1)$$

El programa que vamos a ver a continuación nos muestra cómo usar esta sentencia.

```

1000 REM *****
1010 REM * CASA CON RUEDAS *
1020 REM *****
1030 REM
1040 REM *****
1050 REM * PROGRAMA DEMOSTRACION PARA VER EL FUNCIONAMIENTO *
1060 REM * DE LA INSTRUCCION GET EN EL IBM Y COMPATIBLES *
1070 REM *****
1080 REM
1090 REM *** INICIALIZACION ***
1100 REM
1110 SCREEN 1
1120 CLS
1130 OPTION BASE 0
1140 KEY OFF
1150 REM
1160 REM *** DIBUJO DE LA CASA CON RUEDAS ***
1170 REM
1180 LINE (0,20)-(20,0)
1190 LINE -(40,20)
1200 LINE -(0,20)
1210 LINE -(0,60)
1220 LINE -(40,60)
1230 LINE -(40,20)
1240 LINE (5,25)-(15,35),1,BF
1250 LINE (35,25)-(25,35),1,BF
1260 LINE (15,40)-(25,60),2,B
1270 CIRCLE (10,65),5
1280 CIRCLE (30,65),5
1290 REM
1300 REM *** LECTURA DE EL RECTANGULO EN EL CUAL ESTA LA CASA ***
1310 REM
1320 LET X1=0
1330 LET Y1=0
1340 LET X2=40
1350 LET Y2=70
1360 DIM AX(3+INT((X2-X1)/4)*((Y2-Y1)-1))
1370 GET (X1,Y1)-(X2,Y2),AX
1380 REM
1390 REM *****
1400 REM * PROGRAMA PRINCIPAL *
1410 REM *****
1420 REM
1430 CLS
1440 PRINT " Ahora puedes mover la casa con las siguientes teclas"
1450 PRINT
1460 PRINT "      P = DERECHA"
1470 PRINT "      O = IZQUIERDA"
1480 PRINT "      Q = ARRIBA"
1490 PRINT "      A = ABAJO"
1500 PRINT
1510 PRINT "PULSA UNA TECLA"
1520 LET A$=INPUT$(1)
1530 CLS
1540 LET X=0
1550 LET Y=0
1560 LET XO=0
1570 LET YO=0
1580 LET SX=4
1590 LET SY=4
1600 PUT (X,Y),AX
1610 REM
1620 REM *** MOVIMIENTO DE LA CASA CON RUEDAS ***

```

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```
1630 REM
1640 LET A$=INKEY$
1650 IF A$="P" THEN LET X=X+SX
1660 IF A$="O" THEN LET X=X-SX
1670 IF A$="Q" THEN LET Y=Y-SY
1680 IF A$="A" THEN LET Y=Y+SY
1690 IF X<0 THEN X=0
1700 IF X>270 THEN X=270
1710 IF Y<0 THEN Y=0
1720 IF Y>130 THEN Y=130
1730 PUT (XO,YO),A%
1740 PUT (X,Y),A%
1750 LET XO=X
1760 LET YO=Y
1770 IF A$=" " THEN GOTO 1790
1780 GOTO 1640
1790 REM
1800 REM *** FIN DEL PROGRAMA ***
1810 REM
1820 CLS
1830 FOR I=0 TO 1
1840   FOR J=0 TO 6
1850     PUT(J*45,I*80),A%
1860   NEXT J
1870 NEXT I
1880 REM
1890 REM *** BUCLE INFINITO. PULSE CONTROL Y BREAK PARA TERMINAR ***
1900 REM
1910 GOTO 1910
```

Como podemos ver en el programa, primero dibujamos una «casa» con ruedas y la almacenamos en el vector A%. Después de hecho esto nos aparece una pantalla que nos indica qué teclas tenemos que usar para mover la casa por la pantalla. Cuando nos cansamos de hacerlo, si pulsamos la barra espaciadora, nos aparecerá la casa repetida 14 veces en la pantalla.

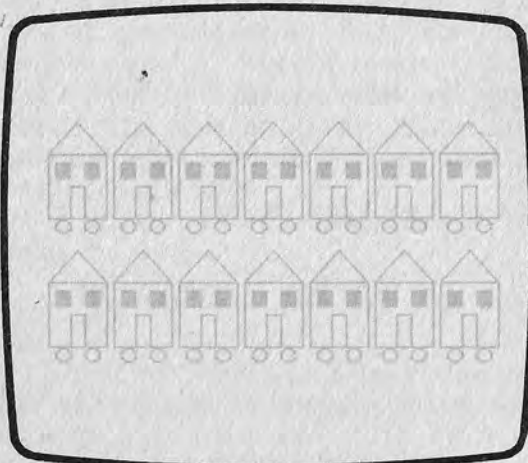


Fig. 2. Ejecución del programa 1.

En el programa aparece una nueva instrucción que no hemos visto todavía. Es la instrucción PUT. Con ella podremos colocar el

gráfico que hemos almacenado con la sentencia GET en cualquier parte de la pantalla.

La instrucción PUT

Esta nueva instrucción será la que nos permitirá colocar el gráfico que hemos almacenado con la instrucción GET en cualquier parte de la pantalla. Su sintaxis es muy parecida a la de la sentencia GET, pero tiene una modificación muy importante:

PUT (X, Y), var, operación

donde:

X = Coordenada en X de la pantalla donde queremos que se coloque la esquina superior izquierda del gráfico.

Y = Coordenada en Y de la pantalla donde queremos que se coloque la esquina superior izquierda del gráfico.

var = Variable (vector) que tiene almacenado el gráfico a poner en pantalla.

operación = Operación lógica que queremos que hagan los puntos del gráfico con los de la pantalla.

Lo más importante de la sentencia PUT, aparte de que nos permite poner gráficos en cualquier parte de la pantalla, es que pode-

mos elegir la operación que queremos que realicen los puntos de dicho gráfico con los puntos que ya hubiese en la pantalla. Las operaciones disponibles son las siguientes:

PSET = Pone los puntos del gráfico que estén iluminados y borra los que estén apagados. Esto significa que, si previamente teníamos algo dibujado en la posición donde vamos a colocar el gráfico, se borrará y aparecerá nuestro gráfico.

PRESET = Realiza la misma operación que PSET, pero invirtiendo los puntos. Si un punto del gráfico original estaba encendido, con este tipo de operación aparecerá apagado. Si estuviese apagado, aparecerá encendido.

AND = Realiza la operación AND entre los puntos del gráfico y los de la pantalla. Esto significa que sólo se imprimirán aquellos puntos que existan en la misma posición en la pantalla y en el gráfico.

OR = Realiza la operación OR entre los puntos de la pantalla y los del gráfico. Esto es, un punto se imprimirá si está en la pantalla, si está en el gráfico o si está en ambos.

XOR = Realiza la operación XOR entre los puntos de la pantalla y los del gráfico. Esta es la opción por defecto si no ponemos ninguna operación en la sentencia PUT. Su efecto es encender los puntos que se encuentren en la

pantalla o en el gráfico y borrar los que se encuentren en la pantalla y en el gráfico.

Para entender mejor el sentido de estas operaciones, en la figura 3 aparece una lista con las tablas de la verdad de todas estas operaciones.

	PSET	PRESET	AND	OR	XOR
1	1	0	1	1	0
0	0	1	0	1	1
1	0	1	1	0	1

Fig. 3. Tablas de la verdad de las operaciones que podemos realizar con el comando PUT.

A continuación vamos a ver otro programa que utiliza unas cuantas sentencias PUT y GET para comprender mejor su funcionamiento.

```

1000 REM *****
1010 REM * CARAS *
1020 REM *****
1030 REM
1040 REM
1050 REM *****
1060 REM * PROGRAMA DEMO PARA VER EL USO DE LAS *
1070 REM * SENTENCIAS GET Y PUT EN EL IBM. *
1080 REM *****
1090 REM
1100 REM *** INICIALIZACION ***
1110 REM
1120 SCREEN 1
1130 CLS
1140 OPTION BASE 0
1150 DIM CX(227)
1160 DIM OCX(10)
1170 DIM OAX(9)
1180 DIM NGX(13)
1190 DIM NPX(13)
1200 DIM CSX(77)
1210 DIM CEX(77)
1220 DIM CNX(77)
1230 REM

```

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```
1240 REM *****
1250 REM * DIBUJO Y ALMACENAMIENTO DE LAS DISTINTAS CARAS *
1260 REM *****
1270 REM
1280 REM *** CABEZA ***
1290 REM
1300 CIRCLE (20,20),20
1310 PAINT (20,20),3,3
1320 GET(0,0)-(40,40),C%
1330 REM
1340 REM *** OJO CERRADO ***
1350 REM
1360 CIRCLE (12,12),3,0
1370 CIRCLE (28,12),3,0
1380 GET(9,9)-(16,16),OC%
1390 REM
1400 REM *** OJO ABIERTO ***
1410 REM
1420 PAINT (12,12),0,0
1430 PAINT (28,12),0,0
1440 CIRCLE (12,12),1,2
1450 CIRCLE (28,12),1,2
1460 GET(9,9)-(16,16),OA%
1470 REM
1480 REM *** NARIZ PEQUENA ***
1490 REM
1500 LINE (18,18)-(22,22),2,BF
1510 GET(16,18)-(24,25),NP%
1520 REM
1530 REM *** NARIZ GRANDE ***
1540 REM
1550 LINE (18,18)-(22,25),2,BF
1560 LINE (16,23)-(24,25),2,BF
1570 GET(16,18)-(24,25),NG%
1580 REM
1590 REM *** CARA SONRIENDO ***
1600 REM
1610 CIRCLE (20,25),10,0,3,14,0
1620 CIRCLE (20,22),10,0,3,14,0
1630 PAINT (20,32),0,0
1640 GET(5,22)-(35,40),CS%
1650 REM
1660 REM *** CARA NORMAL ***
1670 REM
1680 CLS
1690 CIRCLE (20,20),20,3
1700 PAINT (20,20),3,3
1710 LINE (18,18)-(22,25),2,BF
1720 LINE (16,23)-(24,25),2,BF
1730 LINE (10,30)-(30,33),0,BF
1740 GET(5,22)-(35,40),CN%
1750 REM
1760 REM *** CARA ENFADADO ***
1770 REM
1780 CLS
1790 CIRCLE (20,20),20,3
1800 PAINT (20,20),3,3
1810 LINE (18,18)-(22,25),2,BF
1820 LINE (16,23)-(24,25),2,BF
1830 CIRCLE (20,35),10,0,4,2,74
1840 CIRCLE (20,38),10,0,45,2,79
1850 CIRCLE (20,37),10,0,5,2,64
```

```

1860 GET(5,22)-(35,40),CE%
1870 REM
1880 REM *****
1890 REM * PROGRAMA PRINCIPAL *
1900 REM *****
1910 REM
1920 CLS
1930 LOCATE 24,1
1940 PRINT "FIJATE EN LA CARA";
1950 LOCATE 8,15
1960 PRINT CHR$(201);STRING$(10,205);CHR$(187)
1970 FOR I=9 TO 17
1980     LOCATE I,15
1990     PRINT CHR$(186);TAB(26);CHR$(186)
2000 NEXT I
2010 LOCATE 18,15
2020 PRINT CHR$(200);STRING$(10,205);CHR$(186)
2030 LET X=140
2040 LET Y=80
2050 PUT(X,Y),C%
2060 PUT(X+9,Y+9),OA%,PSET
2070 PUT(X+25,Y+9),OA%,PSET
2080 PUT(X+16,Y+18),NG%,PSET
2090 PUT(X+5,Y+22),CS%,PSET
2100 RANDOMIZE TIMER
2110 LET NUM=INT(RND*9)+1
2120 ON NUM GOSUB 2160,2230,2300,2370,2440,2510,2580,2650,2720
2130 FOR I=1 TO 500
2140 NEXT I
2150 GOTO 2110
2160 REM
2170 REM *** OJO DERECHO CERRADO ***
2180 REM
2190 GOSUB 2790
2200 PRINT "CIERRO EL OJO DERECHO"
2210 PUT (X+25,Y+9),OC%,PSET
2220 RETURN
2230 REM
2240 REM *** OJO IZQUIERDO CERRADO ***
2250 REM
2260 GOSUB 2790
2270 PRINT "CIERRO EL OJO IZQUIERDO!"
2280 PUT (X+9,Y+9),OC%,PSET
2290 RETURN
2300 REM
2310 REM *** OJO DERECHO ABIERTO ***
2320 REM
2330 GOSUB 2790
2340 PRINT "ABRO EL OJO DERECHO"
2350 PUT (X+25,Y+9),OA%,PSET
2360 RETURN
2370 REM
2380 REM *** OJO IZQUIERDO ABIERTO ***
2390 REM
2400 GOSUB 2790
2410 PRINT "ABRO OJO IZQUIERDO"
2420 PUT(X+9,Y+9),OA%,PSET
2430 RETURN
2440 REM
2450 REM *** NARIZ GRANDE ***
2460 REM
2470 GOSUB 2790
2480 PRINT "TENGO LA NARIZ GRANDE ***

```

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```

2490 PUT (X+16,Y+18),NG%,PSET
2500 RETURN
2510 REM
2520 REM *** NARIZ PEQUENA ***
2530 REM
2540 GOSUB 2790
2550 PRINT "TENGO LA NARIZ PEQUENA"
2560 PUT(X+16,Y+18),NP%,PSET
2570 RETURN
2580 REM
2590 REM *** BOCA SONRIENDO ***
2600 REM
2610 GOSUB 2790
2620 PRINT "ESTOY CONTENTO"
2630 PUT (X+5,Y+22),CS%,PSET
2640 RETURN
2650 REM
2660 REM *** BOCA NORMAL ***
2670 REM
2680 GOSUB 2790
2690 PRINT "ESTOY PENSATIVO"
2700 PUT (X+5,Y+22),CN%,PSET
2710 RETURN
2720 REM
2730 REM *** BOCA ENFADADA ***
2740 REM
2750 GOSUB 2790
2760 PRINT "ESTOY ENFADADO"
2770 PUT (X+5,Y+22),CE%,PSET
2780 RETURN
2790 REM
2800 REM *****
2810 REM * LIMPIA LA LINEA SUPERIOR DE LA PANTALLA *
2820 REM *****
2830 REM
2840 LOCATE 1,1
2850 PRINT SPACE$(40)
2860 LOCATE 1,1
2870 BEEP
2880 RETURN

```

Las figuras 4 y 5 nos muestran algunas de las caras que es capaz de poner el hombrucillo. El programa se ha realizado de manera que el cambio de expresión sea automá-

tico, pero el lector podría cambiarlo con mucha facilidad para que la expresión cambie al pulsar una secuencia de teclas que él elija.



Fig. 4.



Fig. 5.

La instrucción POINT

Esta nueva instrucción es muy sencilla y muy útil. La utilizaremos para saber de qué color es un punto que se encuentre en una posición X, Y de la pantalla. Su sintaxis es la siguiente:

LET var = POINT (X, Y)

donde:

var = A una variable numérica. Como el resultado de esta instrucción es un número, se puede almacenar en una variable, imprimirlo en pantalla o impresora, operar con él, etc.

X = Coordenada en X del punto que queremos testear.

Y = Coordenada en Y del punto que queremos testear.

Esta instrucción nos dice el color del punto que se encuentra en las coordenadas X, Y. Si dicho punto no está iluminado, entonces nos devolverá un cero (0). Si el punto estuviese iluminado, nos devolverá un número entre 1 y 3, dependiendo del color. Cuando nos encontramos en SCREEN 2, como sólo tenemos un color de tinta, si el punto está iluminado, nos dará el valor uno (1).

La instrucción VIEW

Una de las instrucciones más potentes y útiles del GWBASIC es la instrucción VIEW. Con ella podemos definir una ventana gráfica que ocupe sólo una parte de la pantalla y dibujar en ella sin riesgo de salirnos. Si creamos una pantalla gráfica con el comando VIEW, al realizar un CLS sólo se borrará dicha parte de la pantalla.

La sintaxis del comando VIEW es la siguiente:

VIEW (X1, Y1) - X2, Y2)

donde:

X1 = Coordenada en X de la esquina superior izquierda de la ventana.

Y1 = Coordenada en Y de la esquina superior izquierda de la ventana.

X2 = Coordenada en X de la esquina inferior derecha de la ventana.

Y2 = Coordenada en Y de la esquina inferior derecha de la ventana.

Como puede apreciarse, la sintaxis es muy parecida a la de la sentencia GET, aunque su función es totalmente distinta.

El programa 3, que aparece a continuación, nos muestra algunas de las cosas que podemos hacer con VIEW:

```
1000 REM *****
1010 REM * VENTANAS GRAFICAS *
1020 REM *****
1030 REM
1040 REM *****
1050 REM * PROGRAMA DE DEMOSTRACION PARA VER EL *
1060 REM * USO DE LA SENTENCIA VIEW EN EL IBM. *
1070 REM *****
1080 REM
1090 REM *** INICIALIZACION ***
1100 REM
1110 SCREEN 1
1120 CLS
1130 KEY OFF
1140 REM *** VENTANA No. 1 ***
1150 REM
1160 GOSUB 1680
1170 VIEW(0,0)-(100,100)
1180 LINE (0,0)-(99,99),0,B
1190 GOSUB 1580
1200 GOSUB 1430
1210 GOSUB 1580
1220 REM
1230 REM *** VENTANA No. 2 ***
1240 REM
1250 GOSUB 1680
1260 VIEW (100,100)-(199,199)
```

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```
1270 LINE (0,0)-(99,99),0,B
1280 GOSUB 1560
1290 GOSUB 1430
1300 GOSUB 1560
1310 REM
1320 REM *** VENTANA No. 3 ***
1330 REM
1340 GOSUB 1680
1350 VIEW (40,30)-(139,129)
1360 LINE (0,0)-(99,99),0,B
1370 GOSUB 1560
1380 GOSUB 1430
1390 GOSUB 1560
1400 VIEW
1410 CLS
1420 END
1430 REM
1440 REM *****
1450 REM * DIBUJA *
1460 REM *****
1470 REM
1480 PSET (0,0)
1490 FOR I=0 TO 100 STEP 3
1500   LINE -(100-I,I)
1510   LINE -(100-I,100-I)
1520   LINE -(I,100-I)
1530   LINE -(I,I)
1540 NEXT I
1550 RETURN
1560 REM
1570 REM *****
1580 REM * PULSA UNA TECLA Y BORRA VENTANA *
1590 REM *****
1600 REM
1610 LOCATE 24,1
1620 PRINT "PULSA UNA TECLA PARA BORRAR";
1630 LET A$=INPUT$(1)
1640 CLS
1650 FOR I=1 TO 600
1660 NEXT I
1670 RETURN
1680 REM
1690 REM *****
1700 REM * COLOREAR LA PANTALLA *
1710 REM *****
1720 REM
1730 VIEW : REM TODA LA PANTALLA COMO UNA UNICA VENTANA
1740 CLS
1750 FOR I=1 TO 23
1760   PRINT STRING$(40,219);
1770 NEXT I
1780 RETURN
```

Como hemos visto, la instrucción CLS sólo afecta a la ventana que acabamos de crear. Por otro lado, se puede apreciar que las coordenadas, dentro de dicha ventana, cambian. Ya no son las coordenadas que eran anteriormente, sino que la esquina superior izquierda corresponde a X=0 e Y=0. Si deseamos que no se pierdan las coordenadas origi-

nales, podemos variar la sintaxis de la instrucción VIEW y ponerla como:

VIEW SCREEN (X1, Y1) - (X2, Y2)

Con ello permanecerán las coordenadas originales.

Si ponemos solamente la orden VIEW, sin especificar las coordenadas, lo que hace-

mos es que toda la pantalla sea considerada como una única ventana. Poniéndolo así volvemos a las condiciones originales antes de escribir la primera orden VIEW con coordenadas. Esta orden equivale a poner:

VIEW (0,0) - (319, 199)

La instrucción WINDOW

Pero no acaba aquí lo que podemos hacer con las ventanas que definamos, sino que, además, podemos hacer que el centro de coordenadas se encuentre en el centro del rectángulo que define la ventana. Con ello podemos definir un sistema de coordenadas totalmente independientes de las coordenadas físicas de la pantalla. Su sintaxis es la siguiente:

WINDOW (SX1, SY1) - (SX2, SY2)

donde:

SX1 = uno (1) o menos uno (-1), dependiendo del valor que queremos tener a la izquierda del centro de la ventana. Si es un 1, será positivo, y si es -1, será negativo.

SY1 = uno (1) o menos uno (-1), dependiendo del valor que queremos tener bajo el centro de la ventana. Si es 1, será positivo, y si es -1, será negativo.

SX2 = uno (1) o menos 1 (-1), dependiendo del valor que queremos tener a la derecha del centro de la ventana. Si vale 1, será positivo, y si vale -1, será negativo.

SY2 = uno (1) o menos 1 (-1), dependiendo del valor que queremos tener sobre el centro de la ventana. Si vale 1, será positivo, y si vale -1, será negativo.

Si, por ejemplo, decimos que:

WINDOW (-1,1) - (1, -1)

estaremos definiendo la ventana con un eje de referencia puramente cartesiano, tal y como aparece en la figura 6.

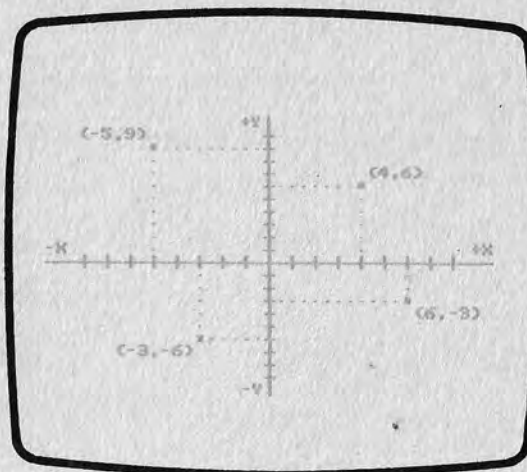


Fig. 6. Eje de coordenadas cartesiano.

mientras que si ponemos:

WINDOW (1,1) - (-1, -1)

el sistema de coordenadas que tendremos será como el que aparece en la figura 7.

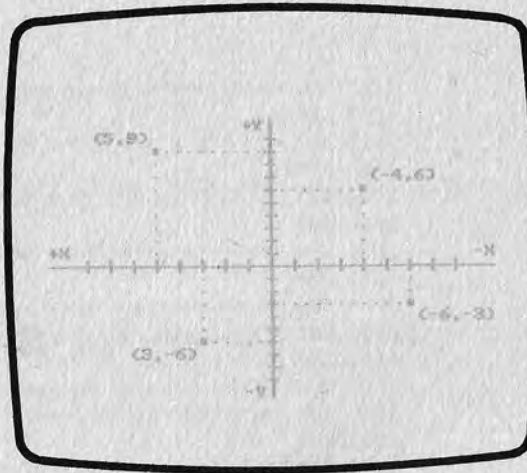
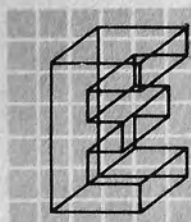


Fig. 7. Eje de coordenadas no cartesiano.

Como ocurría al utilizar la instrucción VIEW cuando queríamos que las coordenadas no cambiasen, sino que siguiesen siendo las físicas de la pantalla, al utilizar el comando WINDOW podemos hacer lo mismo con sólo poner la palabra SCREEN tras la palabra WINDOW de la siguiente manera:

WINDOW SCREEN (SX1, SY1) - (SX2, SY2)

TRUCOS Y RUTINAS BASICAS



El lector puede recordar que durante una serie de tomos estuvimos viendo unos programas que nos permitían definir caracteres y SPRITES en cada uno de los distintos ordenadores que en esta sección tratamos. Debido

a que el IBM no permite redefinir sus caracteres, no incluimos ningún programa para IBM.

Ahora sabemos que podemos definir gráficos y moverlos por la pantalla mediante las instrucciones GET y PUT, por ello, a continuación damos un programa para poder definir y almacenar dichos gráficos.

```
1000 REM *****
1010 REM *
1020 REM *   D E F I N I D O R   D E   G R A F I C O S   *
1030 REM *
1040 REM *           P A R A   I . B . M .           *
1050 REM *
1060 REM *****
1070 REM
1080 REM *****
1090 REM * Programa escrito por: *
1100 REM *
1110 REM * Fco. Morales Guerrero *
1120 REM *****
1130 REM
1140 REM *****
1150 REM *
1160 REM * (c) Ediciones Siglo Cultural *
1170 REM *
1180 REM * (c) 1987.
1190 REM *
1200 REM *****
1210 REM
1220 REM *****
1230 REM *** INSTRUCCIONES ***
1240 REM *****
1250 REM
1260 KEY OFF
1270 SCREEN 2
1280 WIDTH 80
1290 CLS
1300 PRINT TAB(30);"DEFINIDOR DE GRAFICOS"
1310 PRINT TAB(29);"=====
1320 PRINT
1330 PRINT
```

```

1340 PRINT " Usa las teclas del cursor (8,2,4,6) para moverte por la parrilla."
1350 PRINT
1360 PRINT " Pulsa la barra espaciadora para encender o apagar un punto."
1370 PRINT
1380 PRINT " Pulsa F1 para grabar lo definido en formato interno."
1390 PRINT
1400 PRINT " Pulsa F2 para grabar lo definido como lineas DATAS."
1410 PRINT
1420 PRINT " Pulsa F3 para leer un grafico ya definido desde disco."
1430 PRINT
1440 PRINT " Pulsa F4 para imprimir el grafico como lineas DATA."
1450 PRINT
1460 PRINT " Pulsa F9 para salir del programa."
1470 PRINT
1480 PRINT " Solo puede definirse un grafico cada vez."
1490 PRINT
1500 PRINT
1510 PRINT " PULSA UNA TECLA PARA COMENZAR"
1520 LET A$=INKEY$
1530 IF A$="" THEN GOTO 1520
1540 REM
1550 REM *****
1560 REM *** ENTRADA DE DATOS ***
1570 REM *****
1580 REM
1590 CLS
1600 PRINT TAB(30);"DEFINIDOR DE GRAFICOS"
1610 PRINT TAB(29);"=====
1620 PRINT
1630 PRINT
1640 PRINT " Cual va a ser la altura del grafico? (Maximo 19)."
1650 PRINT
1660 INPUT " ==> ",H
1670 IF H<1 OR H>19 THEN PRINT:PRINT "VALOR NO VALIDO":GOTO 1650
1680 PRINT
1690 PRINT
1700 PRINT " Cual va a se la anchura del grafico? (Maximo 40)."
1710 PRINT
1720 INPUT " ==> ",L
1730 IF L<1 OR L>40 THEN PRINT:PRINT "VALOR NO VALIDO":GOTO 1710
1740 REM
1750 REM *****
1760 REM *** INICIALIZACION ***
1770 REM *****
1780 REM
1790 KEY 1,""
1800 FOR I=1 TO 9
1810 KEY I,""
1820 NEXT I
1830 ON KEY(1) GOSUB 2570
1840 ON KEY(2) GOSUB 2840
1850 ON KEY(3) GOSUB 3220
1860 ON KEY(4) GOSUB 3770
1870 ON KEY(9) GOSUB 4150
1880 FOR I=1 TO 9
1890 KEY(I) ON
1900 NEXT I
1910 DIM A%(2*L)
1920 GET(0,0)-(L,H),A%
1930 REM
1940 REM *****
1950 REM *** DIBUJO DE LA PARRILLA Y DE LA PANTALLA ***
1960 REM *****

```

TRUCOS Y RUTINAS BASICAS

```

1970 REM
1980 CLS
1990 FOR I=1 TO H
2000     FOR J=1 TO L
2010         PRINT CHR$(254);
2020     NEXT J
2030     PRINT
2040 NEXT I
2050 COLOR ,,1
2060 LINE(0,183)-(639,191),,BF
2070 LOCATE 24,1
2080 PRINT "F1---SAVE FORMATO INTERNO  F2---SAVE COMO DATA  F3---LOAD  F4---PRI
NT  F9---FIN";
2090 COLOR ,,0
2100 LINE (400,0)-(410+L,10+H),,B
2110 LET X1=405
2120 LET Y1=5
2130 LET X2=X1+L
2140 LET Y2=Y1+H
2150 LET X=1
2160 LET Y=1
2170 REM
2180 REM *****
2190 REM *** PROGRAMA PRINCIPAL ***
2200 REM *****
2210 REM
2220 LET A$=INKEY$
2230 IF A$=" " THEN GOTO 2440
2240 LET DX=0
2250 LET DY=0
2260 IF LEN(A$)=2 THEN LET A$=MID$(A$,2,1)
2270 IF A$=CHR$(77) THEN LET DX=1:LET DY=0
2280 IF A$=CHR$(75) THEN LET DX=-1:LET DY=0
2290 IF A$=CHR$(72) THEN LET DY=-1:LET DX=0
2300 IF A$=CHR$(80) THEN LET DY=1:LET DX=0
2310 LET C=POINT(X1+X,Y1+Y)
2320 IF C=1 THEN LOCATE Y,X:PRINT CHR$(219)
2330 IF C=0 THEN LOCATE Y,X:PRINT CHR$(254)
2340 Y=Y+DY
2350 X=X+DX
2360 IF X>L THEN LET X=L
2370 IF X<1 THEN LET X=1
2380 IF Y>H THEN LET Y=H
2390 IF Y<1 THEN LET Y=1
2400 LOCATE Y,X
2410 PRINT " "
2420 GOTO 2220
2430 GET (0,0)-(L,H),A%
2440 REM
2450 REM *****
2460 REM * FIJACION Y BORRADO DE PUNTOS ***
2470 REM *****
2480 REM
2490 LET C=-POINT(X1+X,Y1+Y)
2500 LET C=-NOT C
2510 PSET(X1+X,Y1+Y),C
2520 GET(X1,Y1)-(X2,Y2),A%
2530 IF C=0 THEN LOCATE Y,X:PRINT CHR$(254)
2540 IF C=1 THEN LOCATE Y,X:PRINT CHR$(219)
2550 GOTO 2220
2560 GOTO 2560
2570 REM
2580 REM *****

```

```

2590 REM *** SAVE EN FORMATO INTERNO ***
2600 REM *****
2610 REM
2620 KEY(1) STOP
2630 LINE(0,159)-(639,167),,BF
2640 LOCATE 21,29
2650 COLOR ,,1
2660 PRINT "SAVE EN FORMATO INTERNO"
2670 COLOR ,,0
2680 GOSUB 4470 : REM PETICION DE NOMBRE
2690 LET N$=N$+".GRP"
2700 LOCATE 22,1
2710 PRINT SPACE$(80)
2720 LOCATE 22,1
2730 PRINT "GRABANDO CON FORMATO INTERNO: ";N$
2740 OPEN N$ FOR OUTPUT AS #1
2750 PRINT #1,L
2760 PRINT #1,H
2770 FOR I=0 TO 2*L
2780     PRINT#1,A%(I)
2790 NEXT I
2800 CLOSE #1
2810 GOSUB 4560 : REM FIN DE FUNCION
2820 KEY(1) ON
2830 RETURN
2840 REM
2850 REM *****
2860 REM *** SAVE COMO LINEAS DATA ***
2870 REM *****
2880 REM
2890 KEY(2) STOP
2900 LINE(0,159)-(639,167),,BF
2910 LOCATE 21,28
2920 COLOR ,,1
2930 PRINT "SAVE COMO LINEAS DATA"
2940 COLOR ,,0
2950 GOSUB 4470 : REM PETICION DE NOMBRE
2960 LET N$=N$+".BAS"
2970 OPEN N$ FOR OUTPUT AS #1
2980 LOCATE 22,1
2990 PRINT SPACE$(80)
3000 LOCATE 22,1
3010 INPUT "DESDE QUE LINEA ==> ";LI
3020 LOCATE 22,1
3030 PRINT SPACE$(80)
3040 LOCATE 22,1
3050 PRINT "GRABANDO COMO LINEAS DATA: ";N$
3060 PRINT #1,LI;"DATA";L;",";H
3070 LET LI=LI+10
3080 FOR I=0 TO 2*L STEP 10
3090     PRINT #1,LI;"DATA";
3100     N=8
3110     IF I+9>2*L THEN LET N=2*L-I-1
3120     FOR J=I TO I+N
3130         PRINT #1,A%(J);",";
3140     NEXT J
3150     PRINT #1,A%(J)
3160     LET LI=LI+10
3170 NEXT I
3180 CLOSE #1
3190 GOSUB 4560 : REM OPERACION TERMINADA
3200 KEY(2) ON
3210 RETURN

```

TRUCOS Y RUTINAS BASICAS

```
3220 REM
3230 REM *****
3240 REM *** LOAD EN FORMATO INTERNO ***
3250 REM *****
3260 REM
3270 KEY(3) STOP
3280 LINE(0,159)-(639,167),,BF
3290 LOCATE 21,29
3300 COLOR ,,1
3310 PRINT "LOAD EN FORMATO INTERNO"
3320 COLOR ,,0
3330 GOSUB 4470 : REM PETICION DE NOMBRE
3340 LET N$=N$+".GRP"
3350 LOCATE 22,1
3360 PRINT SPACE$(80)
3370 LOCATE 22,1
3380 PRINT "CARGANDO CON FORMATO INTERNO: "N$
3390 OPEN N$ FOR INPUT AS #1
3400 ERASE A%
3410 INPUT #1,L
3420 INPUT #1,H
3430 DIM A%(2*L)
3440 FOR I=0 TO 2*L
3450     INPUT #1,A%(I)
3460 NEXT I
3470 CLOSE #1
3480 LOCATE 22,1
3490 PRINT SPACE$(80)
3500 LOCATE 22,1
3510 PRINT "ESPERE UN MOMENTO"
3520 FOR I=1 TO 19
3530     LOCATE I,1
3540     PRINT SPACE$(80)
3550 NEXT I
3560 LOCATE 1,1
3570 FOR I=1 TO H
3580     FOR J=1 TO L
3590         PRINT CHR$(254);
3600     NEXT J
3610     PRINT
3620 NEXT I
3630 LINE(400,0)-(410+L,10+H),,B
3640 LET X2=X1+L
3650 LET Y2=Y1+H
3660 PUT(X1,Y1),A%
3670 FOR I=1 TO H
3680     FOR J=1 TO L
3690         IF POINT(X1+J,Y1+I)=1 THEN LOCATE I,J:PRINT CHR$(219)
3700     NEXT J
3710 NEXT I
3720 LET X=1
3730 LET Y=1
3740 GOSUB 4560 : REM FIN DE FUNCION
3750 KEY(3) ON
3760 RETURN
3770 REM
3780 REM *****
3790 REM *** IMPRESION COMO LINEAS DE DATA ***
3800 REM *****
3810 REM
3820 KEY(4) STOP
3830 LINE(0,159)-(639,167),,BF
3840 LOCATE 21,26
```

```

3850 COLOR ,,1
3860 PRINT "IMPRESION COMO LINEAS DE DATA"
3870 COLOR ,,0
3880 LOCATE 22,1
3890 PRINT "COLOQUE LA IMPRESORA Y PULSE ENTER"
3900 LET A$=INKEY$
3910 IF A$<>CHR$(13) THEN GOTO 3900
3920 LOCATE 22,1
3930 PRINT SPACE$(80)
3940 LOCATE 22,1
3950 INPUT "A PARTIR DE QUE LINEA EMPIEZO ==> ",LI
3960 LOCATE 22,1
3970 PRINT SPACE$(80)
3980 LOCATE 22,1
3990 PRINT "IMPRIMIENDO ... "
4000 LPRINT LI;"DATA";L;",";H
4010 LET LI=LI+10
4020 FOR I=0 TO 2*L STEP 10
4030     LPRINT LI;"DATA";
4040     LET N=8
4050     IF I+N>2*L THEN LET N=2*L-I-1
4060     FOR J=I TO I+N
4070         LPRINT A$(J);",";
4080     NEXT J
4090     LPRINT A$(J)
4100     LET LI=LI+10
4110 NEXT I
4120 GOSUB 4560 : REM FIN DE FUNCION
4130 KEY(4) ON
4140 RETURN
4150 REM
4160 REM *****
4170 REM *** SALIDA DEL PROGRAMA ***
4180 REM *****
4190 REM
4200 KEY(9) STOP
4210 LINE(0,159)-(639,167),,BF
4220 LOCATE 21,32
4230 COLOR ,,1
4240 PRINT "SALIR DEL PROGRAMA"
4250 COLOR ,,0
4260 LOCATE 22,1
4270 PRINT "ESTAS SEGURO? (S/N)"
4280 LET A$=INKEY$
4290 IF A$="" THEN GOTO 4280
4300 IF A$="S" OR A$="s" THEN GOTO 4360
4310 IF A$="N" OR A$="n" THEN GOTO 4330
4320 GOTO 4280
4330 GOSUB 4560 : REM FIN DE FUNCION
4340 KEY(9) ON
4350 RETURN
4360 REM
4370 REM *****
4380 REM *** FIN DEL PROGRAMA ***
4390 REM *****
4400 REM
4410 SCREEN 0
4420 PRINT "ADIOS ..."
4430 FOR I=1 TO 10
4440     PRINT
4450 NEXT I
4460 END
4470 REM

```

TRUCOS Y RUTINAS BASICAS

```

4480 REM *****
4490 REM *** PETICION DE NOMBRE ***
4500 REM *****
4510 REM
4520 LOCATE 22,1
4530 INPUT "NOMBRE ( SIN EXTENSION ) ==> ";N$
4540 IF LEN(N$)>8 THEN GOTO 4520
4550 RETURN
4560 REM
4570 REM *****
4580 REM *** FIN DE FUNCION ***
4590 REM *****
4600 REM
4610 LOCATE 21,1
4620 PRINT SPACE$(160)
4630 LINE(0,159)-(639,167),,BF
4640 LOCATE 21,24
4650 COLOR ,,1
4660 PRINT "FUNCION REALIZADA. PULSE UNA TECLA"
4670 COLOR ,,0
4680 LET A$=INKEY$
4690 IF A$="" THEN GOTO 4680
4700 LOCATE 20,1
4710 FOR I=20 TO 22
4720     PRINT SPACE$(80)
4730 NEXT I
4740 RETURN

```

El funcionamiento del programa es muy sencillo y rápido de comprender. Una vez introducido y ejecutado el programa, nos aparecerán unas breves instrucciones por la pantalla que nos permitirán recordar el uso de los comandos de que disponemos.

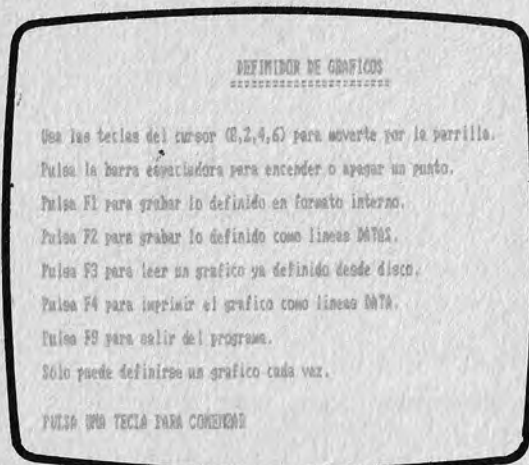


Fig. 1. Resumen de comandos e instrucciones.

A continuación el ordenador nos preguntará el tamaño que tendrá el gráfico a definir. Dicho tamaño está restringido a 19 puntos de largo por 40 de ancho como máximo.

Esto es así porque un gráfico más grande no cabría en la pantalla.

Una vez elegido el tamaño del gráfico que vamos a definir, nos aparecerá la pantalla de diseño:

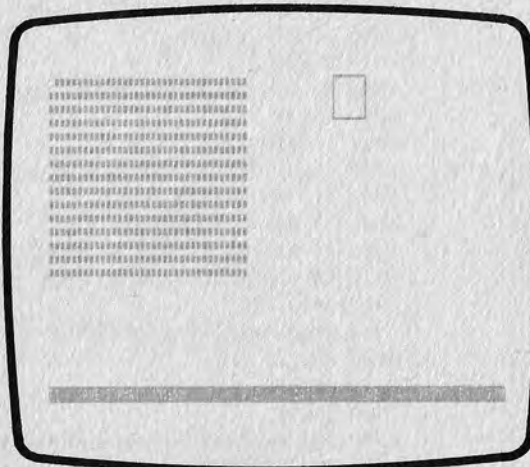


Fig. 2. Una cuadrícula de 19 x 40 antes de empezar a definir.

Podemos distinguir tres zonas distintas:

ZONA 1. Se encuentra en la parte superior izquierda de la pantalla. Nos aparecerá una cuadrícula con tantos puntos verticales y horizontales como queremos que tenga el gráfico a definir.

ZONA 2. Se encuentra en la parte superior derecha de la pantalla. En ella vemos un rectángulo. Dentro del rectángulo iremos viendo, mientras definimos nuestro gráfico, cómo va quedando a tamaño real.

ZONA 3. Ocupa las cuatro últimas líneas de la pantalla y, aparte de informarnos sobre los comandos disponibles, será la zona donde introduciremos los datos que necesite el ordenador.

Para dibujar en la cuadrícula utilizaremos un cursor parpadeante que podremos mover con las teclas del cursor que se encuentran en el teclado numérico. Estas son las siguientes:

- 8- Mover el cursor hacia arriba.
- 2- Mover el cursor hacia abajo.
- 4- Mover el cursor a la izquierda.
- 6- Mover el cursor a la derecha.

Para que dichas teclas funcionen, la luz que se encuentra junto a NUM LOCK ha de estar apagada.

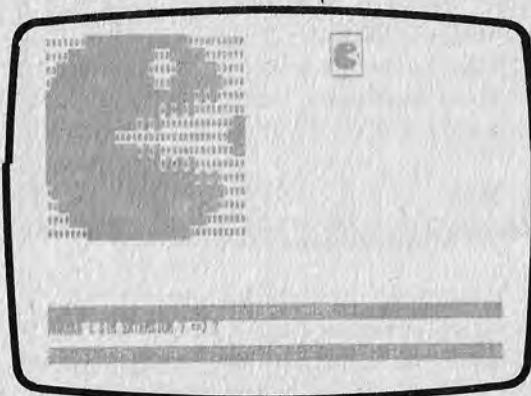


Fig. 3.

Con las teclas del cursor nos movemos dentro de la cuadrícula y, cuando queremos que un punto se ilumine, sólo tenemos que pulsar la barra espaciadora. Si lo que queremos es borrar un punto, entonces también tendremos que pulsar dicha barra. Mientras hacemos esto, nuestro gráfico irá apareciendo en el cuadrado que hay en la ZONA 2.

Una vez que hayamos terminado la definición de nuestro gráfico, podemos grabarlo de dos formas distintas:

Si pulsamos la tecla F1, entonces el gráfico se almacenará en disco de forma que si en otro momento necesitamos redefinirlo o completarlo, usando la opción F3, podemos volver a leerlo.

Si pulsamos la tecla F2, entonces el ordenador nos grabará un fichero con las líneas DATA que contienen la información necesaria para, desde cualquiera de nuestros programas en BASIC, definir el gráfico. El ordenador nos preguntará a partir de qué número de línea empiezan las líneas DATA. El formato de grabación nos permitirá realizar un MERGE de nuestro programa con dichas líneas.

La primera línea DATA sólo contiene dos números. Estos son la altura y la anchura del gráfico, respectivamente. A continuación aparecen una serie de líneas con la definición del gráfico. Dichos números hay que almacenarlos en una matriz entera para poder utilizar el gráfico mediante las instrucciones PUT y GET.

```

1000 REM *****
1010 REM *   PROGRAMA DEMO PARA VER COMO UTILIZAR LAS   *
1020 REM * LINEAS DATAS CON LA DEFINICION DE LOS GRAFICOS *
1030 REM *****
1040 REM
1050 CLS
1060 SCREEN 2
1070 DIM AX(40)
1080 RESTORE 9050
1090 READ L,H
1100 FOR I=0 TO 39
1110   READ AX(I)
1120 NEXT I
1130 LOCATE 14,1
1140 PRINT STRING$(80,219)
1150 FOR I=0 TO 596 STEP 4
1160   FOR J=1 TO 2

```

TRUCOS Y RUTINAS BASICAS

```

1170      PUT(I,93),A%
1180      NEXT J
1190 NEXT I
1200 END
9000 REM
9010 REM *****
9020 REM * LINEAS DATA CON LA DEFINICION DEL GRAFICO *
9030 REM *****
9040 REM
9050 DATA 40 , 10
9060 DATA 41 , 11 , 0 , 0 , 0 , -241 , -1 , 0 , 276 , -32511
9070 DATA 128 , 296 , 24576 , 96 , 336 , 8128 , 255 , 336 , 224 , 1
9080 DATA -16047 , 256 , 193 , 8490 , 512 , -32733 , -8419 , -513 , 220 , -8189
9090 DATA 768 , 224 , -16383 , 256 , 192 , 0 , 0 , 0 , 0 , 0

```

Si pulsamos la tecla F4, el ordenador nos imprimirá un listado de líneas DATA, a partir de la línea que nosotros deseemos, con la definición del gráfico.

Por último, si pulsamos la tecla F9 terminamos el programa.

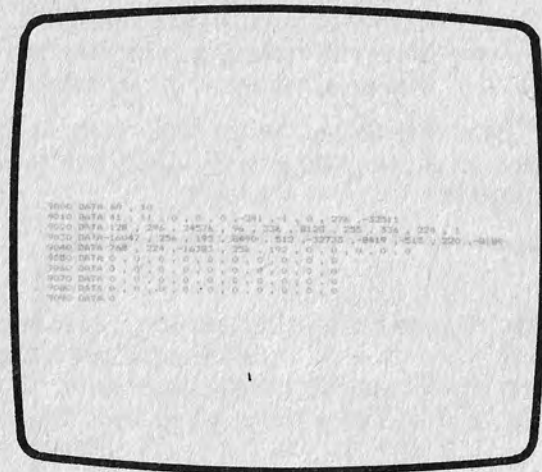
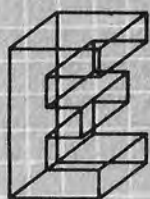


Fig. 4. Ejemplo de impresión al pulsar F4.

EL TALLER DEL HARDWARE

Analizador lógico (segunda parte)



El circuito del analizador lógico permite ser programado desde los bits del puerto de control y los mandos de la carátula externa, para poder seguir las señales del sistema a ensayar. Veamos las diferentes señales que

se producen en el circuito a lo largo de una secuencia de adquisición y algunos programas que permiten controlar la tarjeta y presentar los resultados.

Número de señales muestreadas

El circuito mostrado en el tomo anterior sirve para el registro de 8 señales simultáneas. Puede ampliarse en múltiplos de 8, simplemente replicando la circuitería específica de la adquisición: registro de entrada, memoria, decodificación y circuito de acceso al bus. El resto de los circuitos pueden utilizarse de forma exactamente igual. El circuito de detección de la palabra de disparo deberá también ampliarse para contemplar un ancho de palabra mayor.

El programa de transferencia a memoria del ordenador deberá modificarse en la parte de decodificación, para transferir a un número de vectores igual al de grupos de entradas. La presentación también deberá contemplar distintos paneles para los diferentes grupos de señales.

Señales en el circuito

En la modalidad de uso como analizador de tiempos, se fija una velocidad del reloj

de muestreo suficiente para que puedan registrarse los fenómenos de mínima duración. La tarjeta captura las señales como si de un osciloscopio de muestreo se tratase, pero sin apreciar el nivel de la señal. Únicamente se tomarán las señales como de nivel lógico 1 si superan un umbral fijado en la sonda de entrada. A partir de la sonda todo el proceso es digital. La discretización en el tiempo corresponde al período de muestreo y en los niveles a la tensión de umbral del amplificador de entrada.

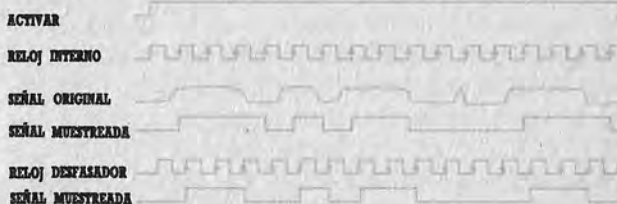


Fig. 1. Señales muestreadas para análisis de tiempos.

Se han mostrado como ejemplo diferentes posibilidades de la señal de entrada, pudiendo observarse el efecto del desfase del pulso de reloj. También queda patente la imposibilidad de registrar un fenómeno transitorio de duración inferior a un pulso de muestreo, a menos que coincida con el flanco del reloj. Otro caso de difícil estimación es el valor que adopta un bus de tres estados cuando todas las salidas están desactivadas. El nivel flotante que resulta puede ser 1 ó 0, según la capacidad del circuito o incluso el tipo de sonda usado. A veces se fuerza ligeramente mediante una resistencia a la tensión de alimentación, para que sea un nivel conocido.

Para que la forma de las señales quedara bien representada sería necesario utilizar un período de muestreo varias veces inferior al menor tiempo a resolver. Los tiempos de su-

EL TALLER DEL HARDWARE

bida de las señales pueden afectar a la forma de onda registrada, si es que son apreciablemente superiores al tiempo de muestreo. De todas maneras, lo que tiene verdadero interés es la relación entre las diferentes señales registradas en el mismo pulso de reloj, para poder determinar la precedencia y las relaciones causa-efecto entre las diferentes unidades.

Un aspecto importante a tener en cuenta es que la carga de las señales en los registros de almacenamiento intermedio y, en general, en cualquier elemento con memoria, requiere de un tiempo de mantenimiento previo a la carga (*settling time*) y otro de mantenimiento después del pulso de carga (*holding time*). Estos tiempos son, en general, de unos pocos nanosegundos para las tecnologías TTL y CMOS, pero pueden ser significativos en algunas aplicaciones.

En la modalidad de uso como analizador de estados el reloj que fija los instantes de muestreo es el del sistema, con algún desfase para ajustarse a las necesidades. El reloj normalmente será periódico, pero podría ocurrir que dentro del ciclo aparecieran subciclos de duración diferente, teniendo entonces que considerar cada muestra como indicación de la existencia de un estado y su posición exacta en el tiempo debería fijarse de acuerdo con las señales capturadas. Esta modalidad permite la realización del muestreo condicionada a la producción de algún fenómeno transitorio y que también afectará a la señal de disparo. Por ejemplo, para seguir la secuencia de estados que se producen al atender la unidad central un fenómeno asíncrono con el proceso, como son las interrupciones, se condicionará el disparo a la señal que produce la interrupción y el reloj a la entrada en una determinada dirección de memoria.



Fig. 2. Señales muestreadas para análisis de estados.

Puede apreciarse que si el reloj no es periódico, el resultado puede diferir aparentemente de la señal original, aunque al ser las muestras del propio sistema se supone que poseerá las transiciones en instantes relacionados.

Si el registro se efectúa utilizando alguna señal para marcado del instante que corresponde al contenido conocido en un conjunto de posiciones del sistema, podría analizarse la secuencia almacenada mediante un programa que convierta cada palabra en el nemotécnico utilizado para la representación de la instrucción o del estado del sistema. Esta es una de las principales ventajas de utilizar un ordenador personal como control de la tarjeta, frente a un equipo específico, por muy adaptado a la resolución del problema que esté.

Una aplicación corriente del analizador consiste en realizar el registro estadístico del número de veces que se ejecuta una determinada secuencia de código. Para ello se condiciona el disparo a las configuraciones que se desea rastrear y se almacena en contadores el número de veces que se produce cada una de ellas. Una representación en diagrama de barras del histograma de frecuencias puede ayudar en la optimización del sistema.

Programas de control de la tarjeta

Para la realización de los análisis de sistemas hemos de contemplar varias partes del programa:

- Inicialización según la aplicación concreta, determinando qué partes del circuito están activas al comienzo y cuál será la secuencia de almacenamiento. Durante la inicialización conviene identificar cada señal para que luego resulte cómoda la interpretación de los resultados. Un menú de definición de señales permitirá hacer el trabajo más fácilmente.

- Captura de las muestras. Generalmente se comienza el almacenamiento al producirse la señal de disparo, pero podría realizarse de forma continua, para detenerse al producirse la condición deseada, teniendo entonces en la memoria la secuencia de estado que condujo al fenómeno.

- Extracción de las muestras tomadas y transferencia a la memoria del ordenador.

- Presentación de los resultados, seleccionando la forma más idónea según la aplicación:

- Secuencia de unos y ceros.
- Palabras completas en notación hexadecimal.

- Formato gráfico como de osciloscopio. Incluirá la posibilidad de compresión, expansión y desplazamientos.

- Formato gráfico bidimensional de estados.

- Formato nemotécnico, si el sistema es conocido y se capturó el contenido del bus de datos.

- Funciones de almacenamiento, restauración, comparación, salida por impresora, etcétera, de señales tomadas en diferentes ensayos.

- Búsqueda de una determinada configuración en los datos almacenados.

Todas las funciones de manipulación de gráficos y ficheros estarán disponibles para hacer más útiles los datos del ensayo y que serán dependientes de la máquina y de las necesidades específicas. Es fundamental que el funcionamiento global del sistema sea cómodo para poder repetir fácilmente cualquier ensayo y verificar que el problema es consistente. Para hacer más rápida la utilización del programa, convendrá usar una versión compilada, pues los ejemplos que se muestran, aunque válidos para hacer la depuración, resultan un poco lentos para aplicaciones reales.

```

10 REM * * * * *
20 REM PROGRAMA DE CONTROL DEL ANALIZADOR LOGICO
50 REM EN EL IBM PC
60 REM * * * * *
61 OPTION BASE 0
62 DEFINT A-Z
63 REM DESCRIPCION DEL PUERTO DE CONTROL:
74 REM * * * * * DEFINICION DE ESTADOS DE CONTROL * * * *
75 REPOSO=&H0: REM NIVELES DE REPOSO AL ARRANCAR
76 ACTIVAR=&H18: REM ACTIVA LA TOMA DE MUESTRAS
77 CONTAR=&H8: REM PERMITE AVANZAR DIRECCION
83 INCREMENTAL=&H0: REM LECTURA CON PULSO DE INCREMENTO
84 INCREMENTAH=&H4
85 INHIBEENTRADA=&H2: REM PARA LECTURA DE LA MEMORIA
86 DIM DATOS(3000)
88 N=100: REM NUMERO DE MUESTRAS
89 F1$="DATOSLA.DAT"
90 REM * * * * * DEFINICION DE PUERTOS
100 PDATOS=&H201: REM DATOS
105 PCONTROL=&H203: REM BYTE DE CONTROL
110 PESTADO=&H203: REM BYTE DE ESTADO
115 KEY OFF
120 CLS
130 PRINT " PROGRAMA DE CONTROL DEL ANALIZADOR LOGICO"
140 PRINT
142 PRINT " VERIFIQUE CONTROLES DE SEÑALES,"
143 PRINT " PALABRA DE DISPARO Y BASE DE TIEMPOS"
149 PRINT
150 PRINT " SELECCIONE OPCION:"
160 PRINT
170 PRINT " 1.-ACTIVAR MUESTREO"
190 PRINT
200 PRINT " 2.-PRESENTAR RESULTADOS"
210 PRINT
220 PRINT " 3.-GUARDAR EN DISCO"
230 PRINT
240 PRINT " 4.-TERMINAR"
260 PRINT
270 INPUT "OPCION=";OPCION
280 IF OPCION<1 OR OPCION>4 THEN BEEP: GOTO 120
290 ON OPCION GOSUB 1000,2000,3000,4000
295 IF OPCION=4 THEN END
300 GOTO 120
990 REM * * * * * ACTIVACION DE TOMA DE MUESTRAS
1000 CLS
1010 PRINT "TOMANDO MUESTRAS"
1170 REM ESTABLECE PARAMETROS DE MUESTREO
1200 OUT PCONTROL,ACTIVAR
1210 FOR I=1 TO 10000
1230 A=INP(PDATOS)
1240 IF A<>0 THEN 1210
1250 NEXT I
1260 PRINT "TERMINO TIEMPO MAXIMO DE TOMA DE MUESTRAS": RETURN
1300 PRINT "FIN DE TOMA DE MUESTRAS"
1305 OUT PCONTROL,REPOSO
1306 OUT PCONTROL,CONTAR
1310 FOR I=1 TO 2048
1320 OUT PCONTROL,INCREMENTAL

```

EL TALLER DEL HARDWARE

```
1330 OUT CONTROL, INCREMENTAH
1340 OUT CONTROL, INCREMENTAL
1350 DATOS(I)=INP(PDATOS)
1360 NEXT I
1500 RETURN
1790 REM * * * * MENU DE PRESENTACION * * *
2000 CLS
2010 REM INCLUIR LOS TIPOS DE PRESENTACION DESEADOS
2020 PRINT "INTEGRAR CODIGO DE PRESENTACION"
2030 FOR I=1 TO N
2040 PRINT HEX$(DATOS(I)); " ";
2050 NEXT I
2090 INPUT "PULSE UNA TECLA PARA SEGUIR"; X$
2100 RETURN
2790 REM * * * * FUNCION DE PUESTA EN REPOSO * * *
3000 REM GUARDA DATOS EN DISCO
3010 OPEN F1$ FOR OUTPUT AS #1
3020 PRINT #1, N
3030 FOR I=1 TO N
3040 PRINT #1, DATOS(I)
3050 NEXT I
3060 CLOSE
3100 RETURN
4000 REM DEJA LA TARJETA EN REPOSO
4010 OUT PCONTROL, REPOSO
4100 RETURN
```

Programa de transferencia de datos

Una vez activado el indicador de que el barrido ha terminado, o transcurrido un tiempo superior al de la toma de muestras en el peor de los casos, se pueden extraer los datos de la memoria exterior y transferirlos a un vector para su tratamiento y presentación. El programa simplemente consistirá en un bucle de incremento de posición del contador exterior y lectura del valor presente en el puerto de entrada. Previamente es necesario desconectar las salidas del registro de entrada que, como son de tres estados, es suficiente con poner el bit de control en 1. El incremento de la dirección de memoria accedida se realiza simulando un pulso de reloj, mediante el correspondiente bit de control. La memoria del almacenamiento intermedio presenta a su salida al bus de datos el contenido almacenado, simplemente con direccionar la posición, no siendo necesario ningún pulso de lectura.

La opción de activación de la toma de muestras, cuando termina, realiza automáticamente la transferencia al vector DATOS.

Programa de presentación de las señales

La observación de las señales capturadas es el objeto de nuestro experimento. Para facilitarlo al máximo deberemos disponer de un programa de presentación que utilice óptimamente las posibilidades gráficas de nues-

tro equipo. El ejemplo que se presenta es una simple muestra de lo que podemos conseguir dedicándole tiempo al problema concreto. Esta es una de las ventajas de utilizar un ordenador: tenemos múltiples soluciones para cada problema, solamente limitadas por nuestra imaginación.

Presentación de las señales como unos y ceros

El programa presenta las muestras como puntos luminosos o de un color tomando como eje horizontal el tiempo, que en este caso será el número de la muestra. En el menú de presentación podríamos seleccionar la muestra inicial a presentar y el número de ellas. En el ejemplo la señal se ha almacenado en un fichero y se lee de él cuando se necesita.

```
10 REM PRESENTACION DE SEÑALES DIGITALES
   EN FICHERO DE DATOS
20 REM T3
30 SCREEN 0,0,0: KEY OFF
40 DIM A(1000)
50 V$(0)="": V$(1)=""
60 F1$="SENO13"
70 FOR I=1 TO 8: M(I)=2(I-1): NEXT I
90 GOSUB 2000: REM LEE LOS DATOS
200 CLS
210 SCREEN 0
220 PRINT " *** REPRESENTACION CON CARAC-
   TERES ***"
230 PRINT " ESCALA DE TIEMPO EN PERIODOS
   DE MUESTREO"
240 REM REJILLA
250 FOR I=6 TO 78 STEP 5
```

```

260 FOR J=5 TO 22 STEP 4
300 LOCATE J,I
302 PRINT " ";
305 NEXT J
310 NEXT I
320 FOR I=5 TO 75 STEP 5
330 LOCATE 4,1: PRINT USING "##" I-5;
340 NEXT I
350 REM DEFINE IDENTIFICACION DE SEÑALES
352 FOR J=1 TO 8
354 LOCATE 4+2*J,2
356 PRINT "X";J;
358 NEXT J
359 REM PRESENTA LAS SEÑALES COMO _Y
360 FOR X=2 TO N
365 XX=INT(255*A(X))
370 FOR J=1 TO 8
384 LOCATE 4+J*2,X+4
386 PRINT V$(SGN(XX AND M(J)));
388 NEXT J
390 NEXT X
420 WHILE INKEY$="": WEND
500 END
2000 REM LECTURA DE DATOS
2020 OPEN "I",#1,F1$
2030 INPUT #1,N
2040 PRINT "DATOS CONTIENE ";N+1;" VALORES"
2070 FOR K=1 TO N
2080 INPUT #1,A(K)
2110 NEXT K
2120 CLOSE
2200 RETURN

```

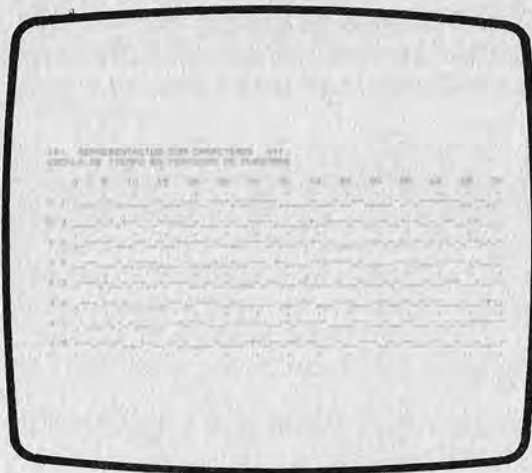


Fig. 3. Salida gráfica como caracteres.

Para poder observar secuencias de muestras más largas, deberemos definir en el programa un valor de comienzo, el cual se añadirá en el índice de selección en la muestra a presentar.

Del programa de presentación mediante caracteres que recuerdan la forma de onda típica de un osciloscopio tiene interés el men-

cionar la forma utilizada para sacar cada bit independientemente. Las sentencias 384-386 indican la forma cómo a partir del elemento del vector que representa a la muestra se obtiene cada bit por separado con la función AND y el elemento de un vector que contiene el bit deseado. El resultado puede ser 0 ó 1. En el primer caso se selecciona el carácter representativo del 0 y en el segundo el del 1. Como la función AND en BASIC da como resultado el valor del bit considerado es necesario extraer el signo, que en ese caso sí que será 0 ó 1.

■ Presentación de señales en modo gráfico

Podemos representar las señales muestreadas como si de un osciloscopio se tratase, utilizando la capacidad gráfica de cada ordenador. El programa que se muestra utiliza el modo de alta resolución del IBM-PC, presentando 8 señales simultáneamente. Cada señal lleva una identificación correspondiente a la entrada por la que se tomó, que debería coincidir con la etiqueta puesta en la sonda.

En el gráfico también se muestran unas marcas, entre cada dos señales, para poder referenciar cada muestra con el instante de muestreo.

```

10 REM PRESENTACION DE SEÑAL EN FICHERO
DE DATOS
20 REM T2
30 SCREEN 0,0,0: KEY OFF
40 DIM A(1000)
50 VRES=50: HRES=600: DESPX1=10: DESPY1=30
60 F1$="SEN13.DAT"
70 FOR I=1 TO 8: M(I)=2(I-1): NEXT I
90 GOSUB 2000: REM LEE LOS DATOS
200 CLS
210 SCREEN 2
220 PRINT " *** REPRESENTACION GRAFICA DE
DATOS DIGITALIZADOS ***"
230 LINE (0,20)-(639,180),1,B
240 FOR I=40 TO 630 STEP 20: REJILLA
250 FOR J=40 TO 160 STEP 32
300 PSET (I,10+J),1
310 PSET (I,11+J),1
320 NEXT J
330 NEXT I
350 REM IDENTIFICACION DE SEÑALES
352 FOR J=1 TO 8
354 LOCATE 4+2*J,2
356 PRINT "X";J;
358 NEXT J
359 REM REPRESENTACION DE LAS SEÑALES
360 FOR X=2 TO N
365 XX=INT(255*A(X))

```

EL TALLER DEL HARDWARE

```
370 FOR J=1 TO 8
384 PSET (40+2*X,30+(J*16)-5*(SGN(XX AND M(J))))
388 NEXT J
390 NEXT X
420 WHILE INKEY$="": WEN
440 SCREEN 0,0,0
500 END
2000 REM LECTURA DE DATOS
2020 OPEN "I",#L,F1$
2030 INPUT #1,N
2040 PRINT "DATOS CONTIENE ";N+1;" VALORES"
2070 FOR K=1 TO N
2080 INPUT #1,A(K)
2110 NEXT K
2120 CLOSE
2200 RETURN
```

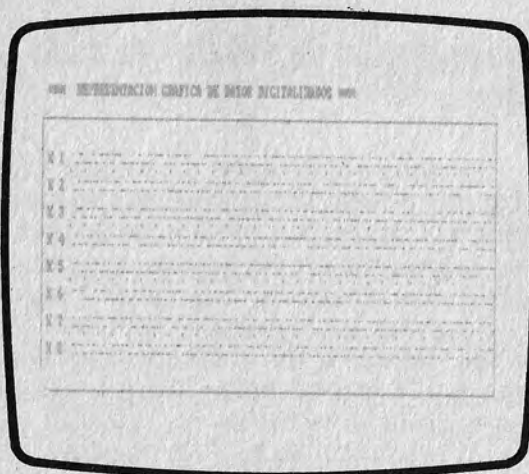


Fig. 4. Salida gráfica.

La ampliación de la señal en la representación gráfica simplemente consistirá en la reproducción del nivel del número de puntos correspondientes a la ampliación. No tendremos más información, pero estará más fácilmente analizable.

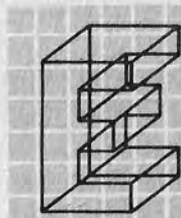
Conclusión

Con el circuito y los programas mostrados pueden ampliarse las posibilidades de nuestro taller de forma importante y con las adaptaciones pertinentes a cada máquina nos permitirá resolver numerosos problemas y afrontar nuevos proyectos con la seguridad de que vamos a poder analizarlos de forma rápida y eficiente.

Es conveniente recalcar que, si bien el sistema será útil para nuestras aplicaciones, queda un poco lejos todavía de la potencia de los equipos comerciales actuales, sin que por ello debamos retraernos de pensar en su montaje. La dificultad es sólo un poco mayor que el montaje de unas cuantas sondas lógicas, con unos resultados muy superiores a su coste.

NATURALEZA Y TECNOLOGÍA

Física: teoría de choques



En este capítulo vamos a tratar la teoría elemental de choque entre dos cuerpos. Es un hecho evidente que no todos los cuerpos de la naturaleza se comportan igual ante un choque. Por ejemplo, es distinta la mane-

ra de chocar entre sí dos bolas de billar que dos automóviles, o la de una mosca contra un tren. Todo esto nos hace pensar que existen diversos factores inherentes a cada cuerpo que les hace comportarse de una manera determinada en un choque con otro objeto.

Choques

Hemos dicho anteriormente que el choque de dos cuerpos depende de diversos factores. Uno de esos factores determinantes es la característica elástica del cuerpo. Durante el choque se producen en ambos cuerpos deformaciones, creándose en el interior de ellos fuerzas elásticas que se oponen a la deformación.

Otro factor importante es la energía cinética. Como ya sabemos, la energía cinética es función de la masa del cuerpo y de su velocidad:

$$E_c = 1/2 \cdot m \cdot v^2$$

Pues bien, si las deformaciones se producen dentro del límite elástico de los cuerpos, éstos recuperan su forma primitiva, devolviendo toda la energía cinética que habían absorbido en el choque. El choque en que esto se verifica se denomina elástico y en él hay conservación de energía cinética. Es el ejem-

plo típico de las bolas de billar, o de una pelota de goma contra el suelo... Si en el choque las deformaciones superan las correspondientes al límite elástico (por ejemplo, el choque de un automóvil contra una pared), la energía necesaria para producir esas deformaciones se consume de la energía cinética en el momento del choque, por lo que la energía cinética no se conserva como en el caso del choque elástico. Este choque se denomina choque inelástico.

En realidad, se presentan todos los casos posibles intermedios entre el choque elástico e inelástico, llamado choque parcialmente elástico, en el que introducimos un parámetro definido como el CR (coeficiente de restitución).

Vamos a realizar un pequeño estudio de cada uno de ellos:

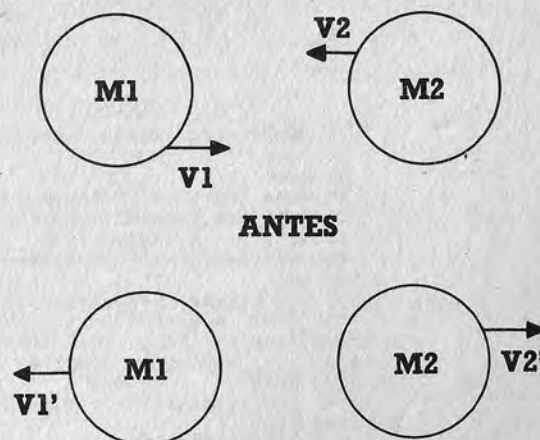


Fig. 1.

A. Choque totalmente elástico

En el choque totalmente elástico se verifica:

APRENDER CON EL ORDENADOR

$$M1 \cdot V1 + M2 \cdot V2 = M1 \cdot V1' + M2 \cdot V2' \quad (1)$$

Que se cumple en cualquier tipo de choques y se denomina conservación de la cantidad de movimiento.

Por ser un choque totalmente elástico también se conserva, como dijimos anteriormente, la energía cinética antes y después del choque:

$$1/2 \cdot M1 \cdot V1^2 + 1/2 \cdot M2 \cdot V2^2 = 1/2 \cdot M1 \cdot V1'^2 + 1/2 \cdot M2 \cdot V2'^2 \quad (2)$$

De las ecuaciones (1) y (2) sacamos despejando las velocidades finales:

$$V1' = \frac{M1-M2}{M1+M2} \cdot V1 + 2 \cdot \frac{M2}{M1+M2} \cdot V2$$

$$V2' = 2 \cdot \frac{M1}{M1+M2} \cdot V1 - \frac{M1-M2}{M1+M2} \cdot V2$$

B. Choque inelástico

Si los cuerpos que chocan se comportan de forma totalmente inelástica, toda la deformación producida en el choque se conserva. Al no haber reacción elástica, los dos cuerpos quedan unidos después del choque, es decir, que:

$$M = M1 + M2$$

$$V' = V1 + V2$$

$$M1 \cdot V1 + M2 \cdot V2 = (M1+M2) \cdot V' \Rightarrow V' = \frac{M1 \cdot V1 + M2 \cdot V2}{M1+M2}$$

y la pérdida de energía cinética es:

$$T = \frac{M1 \cdot M2 \cdot (V1 - V2)^2}{2 \cdot (M1 + M2)}$$

C. Choque parcialmente elástico

Los choques que existen en la naturaleza nunca son totalmente elásticos, pues la energía cinética antes y después del choque es muy difícil que se mantenga debido, sobre todo, a pequeños efectos no deseables, como el rozamiento de la bola de billar contra la mesa, la pequeña cantidad de calor desprendida en el choque o la resistencia del aire al movimiento. Según esto, un choque que consideramos totalmente elástico no será tal, sino será un choque parcialmente elástico con un coeficiente de restitución muy próximo a 1 (CR=1 implica choque totalmente elástico).

En el choque parcialmente elástico se cumple que:

$$V2' - V1' = CR - (V1 - V2)$$

donde $0 < CR < 1$.

Entonces, para un choque totalmente elástico, se verifica que $CR=1$ y en un choque totalmente inelástico $CR=0$.

Y las velocidades finales serían:

$$V1' = \frac{M1-M2 \cdot CR}{M1+M2} \cdot V1 + \frac{M2 \cdot (1+CR)}{M1+M2} \cdot V2$$

$$V2' = \frac{M1 \cdot (1+CR)}{M1+M2} \cdot V1 + \frac{M2-M1 \cdot CR}{M1+M2} \cdot V2$$

Y la variación de energía cinética:

$$T = \frac{(1-CR)^2}{2} \cdot \frac{M1 \cdot M2}{M1+M2} \cdot (V2-V1)^2$$

que para $CR=1$ —choque totalmente elástico—, es nula.

```

10 REM*****
20 REM*      PROGRAMA      *
30 REM*      DE            *
40 REM*      CALCULO DE PARAMETROS *
50 REM*      EN CHOQUES ELASTICOS *
60 REM*      E INELASTICOS *
70 REM*****
80 REM
90 REM *****
100 REM* PROGRAMA VALIDO PARA *
110 REM* AMSTRAD, IBM/COMPATIBLES*
120 REM* SPECTRUM, COMMODORE Y *
130 REM* MSX *
140 REM*****
150 CLS
160 PRINT "ESTE PROGRAMA CALCULA"
170 PRINT "LA MASA, VELOCIDAD Y "
180 PRINT "CANTIDAD DE MOVIMIENTO"
190 PRINT "DESPUES DEL CHOQUE"
200 PRINT "CALCULO DE UN CHOQUE ELASTICO O "
210 INPUT "PARCIALMENTE ELASTICO (E/P) ";H$
220 PRINT :PRINT
230 IF H$<>"E" AND H$<>"P" THEN PRINT "ERROR DE ENTRADA":GOTO 200
240 IF H$="E" THEN GOTO 280
250 INPUT "INTRODUCIR VALOR DEL COEF. RESTITUCION (0<COEF.REST<1)";CR

```

```

260 IF CR>1 AND CR<1 THEN PRINT "COEF. REST. NO VALIDO":GOTO 250
270 GOTO 290
280 LET CR=1
290 INPUT "MASA EN KGR. DEL CUERPO 1":M1
300 INPUT "MASA EN KGR. DEL CUERPO 2":M2
310 INPUT "VELOCIDAD DEL CUERPO 1 (METROS/SEG.)":V1
320 INPUT "VELOCIDAD DEL CUERPO 2 (METROS/SEG.)":V2
330 LET E1=.5*((M1*(V1^2))+(M2*(V2^2)))
340 PRINT "ENERGIA CINETICA ANTES DEL CHOQUE=" ;E1
350 LET AT=((1-CR^2)*M1*M2*(V2-V1)^2)/(2*(M1+M2))
360 LET E2=E1-AT
370 PRINT "ENERGIA CINETICA DESPUES DEL CHOQUE=" ;E2
380 PRINT "PERDIDA DE ENERGIA CINETICA EN EL CHOQUE=" ;AT
390 LET F1=((M1-M2*CR)*V1+(M2*(1+CR))*V2)/(M1+M2)
400 LET F2=((M2-M1*CR)*V2+(M1*(1+CR))*V1)/(M1+M2)
410 IF E=1 THEN GOTO 450
420 PRINT "VELOCIDAD FINAL DEL CUERPO 1=" ;F1
430 PRINT "VELOCIDAD FINAL DEL CUERPO 2=" ;F2
440 END
450 PRINT "LA VELOCIDAD DEL CONJUNTO ES=" ;F1

```

Comentario del programa

La realización del programa es bastante simple. Se distinguen dos zonas bien definidas, la zona de adquisición de datos y la zona de cálculo. Las fórmulas utilizadas son las del choque parcialmente elástico, que dependiendo del coeficiente de restitución se comporta como elástico, inelástico o parcialmente elástico.

Si el coeficiente de restitución es 0, los dos cuerpos permanecen unidos, de ahí que la velocidad sea única.

MATEMATICAS

Cálculo de la matriz traspuesta e inversa

Para la realización de este programa nos hemos basado en el programa de cálculo de determinantes editado anteriormente, y pretendemos completar el capítulo de matrices ya desarrollado en esta colección.

En el campo de los números reales, existe una propiedad conocida por todos, que es la existencia de elemento inverso, esto es, dado un número $a \in \mathbb{R}$, $a \neq 0$, sabemos que existe otro número $b \in \mathbb{R}$, que verifica que:

$$a \cdot b = 1.$$

Está claro que ese número $b = \frac{1}{a}$.

También algunas matrices poseen esta propiedad, pero no todas. Veamos cuál es el conjunto de las matrices que lo cumplen:

Sea A matriz cuadrada, es decir, que tiene el mismo número de filas que de columnas, y sea el determinante de A , $\text{DET}(A) \neq 0$, entonces se verifica que existe una matriz B , también cuadrada, tal que:

$$(A) \cdot (B) = 1$$

Donde la matriz B se obtiene hallando la traspuesta de A y dividiendo cada término por el determinante de A . La demostración de este teorema se sale de nuestras pretensiones, si bien está en cualquier libro elemental de álgebra.

Pero en la definición anterior hemos utilizado el término "traspuesta de A " sin haberlo definido previamente. Veámoslo con un pequeño ejemplo:

$$(A) = \begin{bmatrix} 1 & 2 & 5 \\ 4 & 7 & 9 \\ 0 & 8 & 3 \end{bmatrix} \text{ su traspuesta es } T((A)) = \begin{bmatrix} 1 & 4 & 0 \\ 2 & 7 & 9 \\ 5 & 9 & 3 \end{bmatrix}$$

Es decir, trasponer una matriz es intercambiar sus filas por columnas.

La matriz inversa será, pues:

$$I((A)) = (B) = \begin{bmatrix} 1/|A| & 4/|A| & 0/|A| \\ 2/|A| & 7/|A| & 9/|A| \\ 5/|A| & 9/|A| & 3/|A| \end{bmatrix}$$

La matriz inversa se utiliza muchísimo en álgebra lineal para problemas de cambio de base, de resolución de ecuaciones...

APRENDER CON EL ORDENADOR

```

10 REM *****
20 REM *   CALCULO DE MATRICES INVERSAS   *
30 REM *   VALIDO                          *
40 REM *   PARA IBM-PC, MSX, AMSTRAD,     *
50 REM *   COMMODORE, SPECTRUM            *
60 REM *****
70 REM *****
80 REM * ORDEN DE LA MATRIZ                *
90 REM *****
100 INPUT "ORDEN DE LA MATRIZ (SOLO HASTA CUATRO) ";O
110 IF O>4 OR O<2 THEN PRINT "SOLO HASTA RANGO CUATRO" :GOTO 100
120 DIM X(O,O),Y(O,O),A(O,O),Z(O),T(O,O)
130 REM *****
140 REM * VALORES DEL DETERMINANTE          *
150 REM *****
160 PRINT "INTRODUZCA LOS VALORES DE LA MATRIZ"
170 FOR F=1 TO O
180 FOR C=1 TO O
190 PRINT "FILA ";F;
200 PRINT "COLUMNA ";C;
210 INPUT " ";X(F,C)
220 NEXT C
230 NEXT F
240 IF O>2 THEN GOTO 300
250 REM *****
260 REM * DETERMINANTE DE ORDEN 2          *
270 REM *****
280 LET R=X(1,1)*X(2,2)-X(1,2)*X(2,1)
290 GOTO 670
300 IF O>3 THEN GOTO 430
310 REM *****
320 REM * DETERMINANTE DE ORDEN 3          *
330 REM *****
340 FOR I=1 TO O
350 FOR J=1 TO O
360 LET A(I,J)=X(I,J)
370 NEXT J
380 NEXT I
390 GOSUB 790
400 LET R=P
410 GOTO 670
420 REM *****
430 REM * DETERMINANTE DE ORDEN 4          *
440 REM *****
450 FOR D=1 TO O
460 GOSUB 860
470 FOR F=2 TO O
480 FOR C=1 TO O
490 LET Y(F-1,C)=Y(F,C)
500 NEXT C
510 NEXT F
520 FOR F=1 TO O-1
530 FOR C=1 TO O-1
540 IF D<>O THEN Y(F,C)=Y(F,C+1)
550 NEXT C
560 NEXT F
570 GOSUB 700
580 GOSUB 790
590 LET Z(D)=P
600 NEXT D
610 FOR F=1 TO O
620 LET R=R+(-1)^(1+I)*Z(F)*X(1,F)
630 NEXT F
640 REM *****
650 REM * SALIDA DEL RESULTADO DEL DETERMIN. *
660 REM *****
670 PRINT "EL VALOR DEL DETERMINANTE ES ";R
675 PRINT "PULSA ENTER PARA CONTINUAR"
676 INPUT W$
680 GOTO 940
690 REM *****
700 REM * CALCULO DE ADJUNTOS DE ORDEN 3    *
710 REM * *****                          *
720 FOR I=1 TO O-1
730 FOR J=1 TO O-1
740 LET A(I,J)=Y(I,J)
750 NEXT J
760 NEXT I
770 RETURN
780 REM *****
790 REM * CALCULO DEL DETERMINANTE DE ORDEN 3 *
800 REM *****
810 LET P=A(1,1)*A(2,2)*A(3,3)-A(2,3)*A(3,2)

```

```

820 LET P=P-A(1,2)*(A(2,1)*A(3,3)-A(2,3)*A(3,1))
830 LET P=P+A(1,3)*(A(2,1)*A(3,2)-A(2,2)*A(3,1))
840 RETURN
850 REM *****
860 REM * DETERMINANTE AUXILIAR 2 *
870 REM *****
880 FOR I=1 TO 0
890 FOR J=1 TO 0
900 LET Y(I,J)=X(I,J)
910 NEXT J
920 NEXT I
930 RETURN
940 REM *****
950 REM * CALCULO DE LA TRASPUESTA *
960 REM *****
970 PRINT " LA TRASPUESTA SERA: "
980 PRINT
990 FOR F=1 TO 0
1000 FOR C=1 TO 0
1010 LET T(F,C)=X(C,F)
1020 PRINT "T(";F;",";C;")= ";T(F,C)
1030 NEXT C
1040 NEXT F
1045 PRINT "PULSE ENTER PARA CONTINUAR"
1046 INPUT W$
1050 REM *****
1060 REM * CALCULO DE LA INVERSA *
1070 REM *****
1080 IF R=0 THEN PRINT"NO EXISTE LA INVERSA DE ESTA MATRIZ":GOTO 2000
1090 PRINT "LA INVERSA SERA: "
1100 PRINT
1120 FOR F=1 TO 0
1130 FOR C=1 TO 0
1140 PRINT "I(";F;",";C;")= ";T(F,C)/R
1150 NEXT C
1160 NEXT F
2000 END

```

Comentario del programa

El programa consta de tres partes muy bien definidas:

— *El cálculo del determinante.*

En su día ya se explicó ampliamente el desarrollo de este programa. Basta decir que el determinante mayor que puede calcular es de orden 4. Esto implica que el rango de la matriz que queremos invertir tiene que ser de orden menor o igual que cuatro.

— *La trasposición de la matriz.*

La operación de trasposición de la matriz es sumamente sencilla. Lo que se pretende con los dos bucles FOR..NEXT anidados es intercambiar la posición de las filas y las columnas, por eso asignamos a $T(F,C) = X(C,F)$. Obsérvese que C y F están intercambiadas.

— *El cálculo de la inversa.*

Para calcular la inversa lo único que he-

mos hecho ha sido dividir cada elemento de la matriz por el resultado del determinante, hallado previamente en la primera parte.

SOCIEDAD

Formas de gobierno en el mundo

Este programa es un pequeño test para comprobar sus conocimientos de las distintas formas de gobierno en el mundo. Debido a la gran variedad de sistemas de gobierno, los hemos agrupado en:

- Monarquía parlamentaria.
- Comunismo.
- República.
- República islámica.

No obstante, este programa se puede ampliar, modificar o corregir con nuevos países o nuevas formas de gobierno en los países ya especificados.

```

10 REM*****
20 REM* PROGRAMA *
30 REM* DE *
40 REM* REPASO DE *
50 REM* FORMAS DE GOBIERNO *
60 REM*****

```

APRENDER CON EL ORDENADOR

```
70 REM
80 REM *****
90 REM * VALIDO PARA IBM-PC, MSX *
100 REM* AMSTRAD , COMMODORE *
109 REM* Y SPECTRUM *
110 REM*****
120 PRINT "ESTE PROGRAMA SIRVE PARA COMPROBAR"
130 PRINT "SUS CONOCIMIENTOS SOBRE LOS DIFERENTES"
140 PRINT "PAISES Y SUS RESPECTIVAS FORMAS DE "
150 PRINT "GOBIERNO"
160 DIM G$(20,2)
170 LET P=0
180 FOR A=1 TO 20
190 FOR B=1 TO 2
200 READ G$(A,B)
210 NEXT B
220 NEXT A
230 FOR A=1 TO 20
250 PRINT "QUE FORMA DE GOBIERNO TIENE ";G$(A,1)
260 LINE INPUT R$
270 IF G$(A,2)=R$ THEN PRINT "RESPUESTA ACERTADA":LET P=P+1:GOTO 290
280 PRINT "RESPUESTA INCORRECTA: LA RESPUESTA CORRECTA ES ";G$(A,2)
290 NEXT A
300 CLS
310 PRINT "RESPUESTAS CORECTAS ";P
1000 DATA "ESPAÑA","MONARQUIA PARLAMENTARIA"
1010 DATA "U.R.S.S","COMUNISMO"
1020 DATA "REINO UNIDO","MONARQUIA PARLAMENTARIA"
1030 DATA "MALASIA","MONARQUIA PARLAMENTARIA"
1040 DATA "ITALIA","REPUBLICA"
1050 DATA "CUBA","COMUNISMO"
1060 DATA "REINO UNIDO","MONARQUIA PARLAMENTARIA"
1070 DATA "E.E.U.U.," "REPUBLICA"
1080 DATA "INDIA","REPUBLICA"
1090 DATA "ARABIA","MONARQUIA PARLAMENTARIA"
1100 DATA "R.D.A.," "COMUNISMO"
1110 DATA "COREA DEL NORTE","COMINISMO"
1120 DATA "IRAN","REPUBLICA ISLAMICA"
1130 DATA "ARGENTINA","REPUBLICA"
1140 DATA "SUECIA","MONARQUIA PARLAMENTARIA"
1150 DATA "R.F.A","REPUBLICA"
1160 DATA "BELGICA","MONARQUIA PARLAMENTARIA"
1170 DATA "IRLANDA","REPUBLICA"
1180 DATA "ANGOLA","COMUNISMO"
1190 DATA "JAPON","MONARQUIA PARLAMENTARIA"
```

Comentarios al programa

El programa consiste básicamente en cargar una matriz de datos, la G\$(), con las preguntas y las respuestas. Para ello utilizamos una matriz N x 2, donde N es el número de países, y en cada fila almacenamos el país y la forma de gobierno correcta. Los datos están en el programa almacenados en DATA, y accedemos a ellos a través de READ.

Llevamos también una cuenta de preguntas acertadas en la variable numérica P.

mulo. La importancia de la capacidad de reacción en una persona ante determinadas circunstancias es muchas veces crítica, y este programa comprueba aproximadamente tu capacidad de respuesta.

Cuando ejecutes el programa tendrás que pulsar ENTER (=RETORNO DE CARRO) una vez que estés preparado. En la parte superior de la pantalla se imprimirá:

"PREPARADO,LISTO..."

PARA LOS MAS JOVENES

Programa para comprobar los reflejos

Este programa sirve para poder comprobar tus reflejos ante un determinado estímulo.

En cuanto aparezca "¡YA!!" deberás pulsar lo antes posible la barra de espacio en blanco. Según la rapidez de respuesta el ordenador te puntúa de 0 a 100. Una buena puntuación es más de 80 puntos, regular de 50 a 80, y menos de 50 puntos indican que tus reflejos no andan muy bien.

```

10 REM*****
20 REM*          PROGRAMA          *
30 REM*          DE                *
40 REM* TEST      DE REFLEJOS     *
50 REM*****
60 REM
70 REM*****
80 REM* PROGRAMA VALIDO PARA      *
90 REM* AMSTRAD, IBM Y MSX        *
100 REM*****
110 CLS
120 PRINT "pulsa barra espacio"
130 PRINT "cuando empiecen las estrellas"
140 INPUT " PULSA ENTER CUANDO ESTES LISTO";A$
150 CLS
160 PRINT "PREPARADO, LISTO..."
170 X=(RND(100))*20000
180 FOR N=1 TO X
190 NEXT N
200 LOCATE 1,18:PRINT "¡¡YA!!"
210 H$=INKEY$
220 IF H$=" " THEN GOTO 260
230 C=C+1
240 LOCATE 20,C:PRINT""
250 GOTO 210
260 LOCATE 21,1:PRINT "PUNTOS";100-C
270 REM*****
280 REM* MODIFICACIONES PARA SPECTRUM*
290 REM*****
300 REM
310 REM*****
320 REM* SUSTITUIR LAS SIGUIENTES LINEAS *
330 REM* 200 PRINT AT 1,18;"¡¡YA!!"      *
340 REM* 240 PRINT AT 20,C;"*"          *
350 REM* 260 PRINT AT 21,1;"PUNTOS";100-C*
360 REM*****
370 REM*****
380 REM* MODIFICACIONES PARA COMMODORE *
390 REM*****
400 REM*****
410 REM* 200 LET X=18:LET Y=1            *
420 REM* 201 GO SUB 1000                 *
430 REM* 202 PRINT "¡¡YA!!"              *
440 REM* 240 LET X=C:LET Y=20            *
450 REM* 241 GO SUB 1000                 *
460 REM* 242 PRINT "*"                  *
470 REM* 260 LET X=1:LET Y=21            *
480 REM* 261 GO SUB 100                  *
490 REM* 262 PRINT "PUNTOS ";100-C      *
500 REM* 1000 LET T$=CHR$(19)            *
510 REM* 1010 LET A$=CHR$(17)            *
520 REM* 1020 LET D$=CHR$(29)            *
530 REM* 1030 PRINT T$                  *

```

```

540 REM* 1040 FOR G=1 TO Y              *
550 REM* 1050 PRINT A$                  *
560 REM* 1060 NEXT G                   *
570 REM* 1070 FOR G=1 TO X              *
580 REM* 1080 PRINT D$                  *
590 REM* 1090 NEXT G                   *
600 REM* RETURN                          *
610 REM*****

```

Comentario del programa

Para la realización de este programa hemos utilizado varios trucos, que vamos a explicar ahora.

Uno de los trucos es LET $X=(RND(100))*20000$. Con esto lo que hacemos es que el tiempo que tarda en salir ¡¡YA!! no sea siempre el mismo, sino que varíe. Puede ocurrir, si tienes un ordenador muy rápido (un AT/IBM o similar), que todo ocurra demasiado de prisa, no te preocupes, cambia el valor de "20000" por otro mayor y el tiempo será mayor. Si, por el contrario, tienes un ordenador lento, no tienes más que disminuir ese valor. No obstante, prueba a cambiar ese valor y verás que el tiempo que tarda el ordenador en darte la respuesta es diferente.

El otro de los trucos es utilizar la función INKEY\$. Esta es una función que se utiliza muchísimo en juegos, y lo que hace es leer la tecla que tú estás pulsando. Por eso cuando tú pulsas la tecla de espacio, el ordenador lo detecta y te indica la puntuación que has obtenido.

PEQUEÑA HISTORIA DE LA INFORMATICA

La Segunda Guerra Mundial



IN duda alguna, la Segunda Guerra Mundial, y la guerra fría que se estableció posteriormente entre las dos grandes potencias mundiales, han contribuido notablemente al desarrollo del ordenador, abriendo paso a

una era distinta de las anteriores, la era de la Informática.

Los científicos de todas las épocas han tenido que sufrir para sacar a la luz sus descubrimientos. Bien es cierto que algunos (los menos) disponían de recursos económicos que les permitían poderse dedicar a sus aficiones; pero también es frecuente que, aun habiendo llevado vidas ejemplares (desde el punto de vista científico), y disponiendo en el mejor de los casos de dinero, hayan sufrido incomprendimientos, no se les haya apoyado lo suficiente, o hayan tenido que disponer de sus recursos personales y perderlos poco a poco, para poder perseverar en sus estudios. ¡Y estamos hablando de los más afortunados! Los recursos para investigación siempre han sido escasos. Menos en esta mitad del siglo XX, y en los Estados Unidos de América. La guerra hizo realidad multitud de proyectos que existían en las mentes de sus creadores únicamente a nivel de sueños. Existió financiación suficiente para desarrollar miles de hipótesis. Y además se dieron unas condiciones muy especiales, que contribuirían muy eficazmente al desarrollo de la Informática.

El Departamento de Defensa norteamericano tenía entre su personal a militares, que a la vez eran magníficos ingenieros, y que colaboraron codo con codo con universitarios, investigadores de primera fila, empresarios audaces, etc. Ese fue el verdadero éxito: la ósmosis existente entre Universidad, empresas y Ejército de los Estados Unidos. Como el Ejército necesitaba tecnología muy avanzada, y disponía de recursos, para obtener lo que necesitaba, no dudó en apoyarse en todo tipo de

personal civil (entre los que se encontraban personas de enorme talla).

Entre otras tareas importantes a desarrollar, se vio pronto la necesidad de crear unos enormes calculadores muy potentes, que el Ejército necesitaba principalmente para cálculos de balística, y posteriormente para cálculos relacionados con la energía nuclear, que estaban desarrollando. Aunque estos calculadores se habían diseñado pensando en ser utilizados durante la contienda, la verdad es que la mayor parte se terminaron una vez finalizada la guerra. De este período recordemos a Von Neumann, quizá una de las personas que más contribuyó al progreso del ordenador en esos años, y a los científicos-empresarios Eckert y Mauchly, infatigables, que prosiguieron su camino separándose de su compañero Von Neumann. El ordenador que marcó este período tan fructífero fue el ENIAC.

La balística no era una ciencia simplemente útil. En esos años su estudio era primordial. La guerra se había modernizado, y ya no la ganaba aquel que disponía de más hombres, o de hombres más valientes. Las armas habían comenzado a tener un papel fundamental. Y entre ellas, los proyectiles, de todo tipo. A la vista de la enorme importancia que iba cobrando esta ciencia, el Ejército creó un laboratorio especial de investigación en balística, el Ballistic Research Laboratory, que instaló en una pequeña localidad de Maryland, de nombre escocés: Aberdeen.

Y, como antes hemos indicado, en este Laboratorio no sólo trabajaban militares. Ni siquiera lo dirigía un militar: estaba regido por un consejo formado por cinco personas, de las que dos eran civiles, y las otras tres militares. Naturalmente, estos militares tenían también una magnífica formación científica, proveniente del Massachusetts Institute of Technology, principalmente. El director científico llegó al Centro de otra Universidad prestigiosa, Princeton (en concreto del Institute of Advanced Studies). Von Neumann, relacionado también con Princeton, pertenecerá al consejo científico.

Pero ¿cuál era la tarea principal de este laboratorio? La creación de tablas de balística. Los proyectiles no eran ya simples bolas de piedra, o de metal, como en tiempos de los piratas. Eran muy caros, y resultaba de suma importancia que los artilleros desperdiciaran el mínimo material posible, acertando en el blanco a la primera. El dicho popular «Quien da primero, da dos veces» es sabiduría antigua pero sigue siendo válido con las técnicas modernas.

Las tablas de balística son muy complejas. En ellas se pueden considerar muchos factores, entre los que mencionaremos el tipo de proyectil, su peso, situación del blanco, velocidad del viento, temperatura, etc.). Si sólo se consideran dos de estos factores, los fundamentales, el alcance del proyectil y la altura del blanco, es necesario calcular entre 2.000 y 4.000 trayectorias para obtener un dato fiable para un único grupo proyectil-cañón (la ope-

Sabía usted que...

Hagamos un pequeño resumen biográfico de la figura de Von Neumann.

Nació en Budapest (Hungría) en 1903. A los veintidós años ya se había doctorado en Matemáticas, aunque varios años antes ya escribía artículos sobre temas que le interesaban siempre en el campo de las Matemáticas. Con veinticuatro años pasó a la Universidad de Berlín, con el grado de «privatdozent», cargo que era muy difícil de obtener siendo tan joven. Este período fue muy fructífero, y en él Von Neumann escribió numerosos artículos de álgebra, mecánica cuántica y, en general, se interesó mucho por la resolución matemática de problemas de estrategia.

Con veintisiete años se instala en los Estados Unidos, como lector asociado en la Universidad de Princeton, y es muy lógico que con la creación del Institute of Advanced Studies se le nombrara profesor permanente (al mismo tiempo que al padre de la Mecánica Cuántica, el profesor Einstein.) Von Neumann seguirá unido a esta Universidad, aunque colaborará en proyectos en cierto modo externos a ella hasta 1955, año en el que fue nombrado por el presidente Dwight Eisenhower miembro de la Comisión de Energía Atómica.

Von Neumann ha sido el padre del ordenador actual. La estructura de sus primeras máquinas se ha conservado hasta nuestros días. Sus notables conocimientos matemáticos eran aplicables a numerosos

campos, como, por ejemplo, a los cálculos matemáticos de las reacciones que se llevarían a cabo durante la explosión de la bomba H. Su contribución a este proyecto, en Los Alamos, fue fundamental, ya que gracias a sus cálculos se pudieron detectar errores considerables en el diseño. Pero Von Neumann era un hombre universal, y también se interesó por otras disciplinas menos peligrosas y desagradables. En los últimos años de su vida estaba estudiando el cerebro humano desde distintos puntos de vista (neurofisiológico y psiquiátrico). Negó la suposición de que las neuronas no eran infalibles, demostrando matemáticamente que se podría construir un autó-mata todo lo fiable que fuera necesario, utilizando para ello elementos susceptibles de error (órgano mayoritario de Von Neumann). Todos estos trabajos eran realizados con el fin de reproducir mediante circuitos y de la forma más sencilla y esquemática posible un órgano que la naturaleza había creado con tanta perfección.

Von Neumann era un hombre de trato muy agradable, detallista, refinado, cortés. Su apariencia física no era muy agraciada. Era grande, amplio de frente, ojos oscuros. Murió muy prematuramente, a la edad de cincuenta y cuatro años, de un cáncer de huesos.

ración requería unas 750 multiplicaciones de diez cifras), y repetir la operación para cada posible par proyectil-cañón.

Nuevas tecnologías de guerra se iban desarrollando con rapidez vertiginosa. Pero el Ballistic Research Laboratory no podía ir absorbiendo el nuevo material que iba apareciendo. Y de qué servía disponer de un material moderno y eficaz si no se sabía manejar, y, por tanto, sacar partido de sus ventajas. En ese período el laboratorio estaba absolutamente desbordado. Cada tabla tardaba unos dos meses en construirse, y la «cola» iba aumentando hasta llegar a situaciones desesperadas. En el verano de 1944 la situación llegó a ser insostenible, produciéndose una media de quince tablas por semana, siendo necesarias unas cuarenta.

Sin embargo, el Ballistic Research Laboratory se había preocupado desde bastante antes de este problema, y había creado un pequeño equipo a las órdenes de Vannevar Bush encargándoles desarrollaran un calculador capaz de abordar el problema. Era 1935, y las necesidades no eran acuciantes.

A pesar de estas previsiones, al comienzo de la guerra, en el laboratorio, los cálcu-

Sabía usted que...

Para calcular una única trayectoria de proyectil, un matemático calculista trabajaba durante tres días, poniendo el máximo cuidado y atención. El analizador diferencial del MIT, o el de la Moore School, de la Universidad de Pensilvania (en los Estados Unidos sólo contaban con esas dos máquinas), hubiera tardado entre quince o veinte minutos (aunque al cambiar de trayectoria hubiera sido necesario disponer de unas dos horas para cambiar todas las conexiones y cableados).

Un matemático calculista, trabajando ocho horas diarias en los cálculos, hubiera tardado unos trescientos años en crear una tabla, naturalmente, sin ayudarse de ningún tipo de calculadora. Con algún tipo de calculadora, el tiempo habría reducido muchísimo, ¡nada menos que a doce años!

Para resolver el problema de la creación de la tabla, el ordenador MARK I hubiera tardado un tiempo mínimo de unos seis meses. A partir de este punto, los tiempos se fueron rebajando hasta llegar al mejor del momento: el del analizador diferencial (del que hemos hablado), que sólo tardaría un mes).

los de trayectoria se realizaban manualmente y eran realizados por mujeres, en su mayor parte, probablemente porque fueran más disciplinadas y cuidadosas que los hombres. Se trataba de diplomadas en Matemáticas, que tardaban unos tres días en calcular una sola trayectoria, y así proseguían lenta y cuidadosamente en el cálculo de la tabla. El laboratorio contaba entonces con unas cien mujeres calculistas.

Pero el Ballistic Research Laboratory estaba además muy bien relacionado con varias Universidades, especialmente con las más activas, entre ellas la de Pensilvania y Princeton. En la Universidad de Pensilvania se encontraba la Moore School, fundada en 1923. Era, por tanto, una escuela de ingeniería electrónica muy reciente. Sin embargo, estaba magníficamente considerada. Disponía, además, de un Analizador Diferencial, dispositivo del que sólo existía otra unidad funcionando en el Massachusetts Institute of Technology. Estas dos Universidades eran enormemente activas, financiando proyectos, contratando profesores asociados de primera talla y organizando cursos incluso durante los veranos.

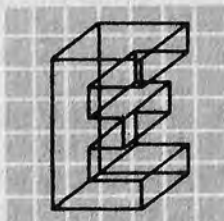
Precisamente en uno de estos cursos de verano, financiado por el Ballistic Research La-

boratory, fue donde Hermann Goldstine, uno de los directivos del BRL, se puso en contacto con John Mauchly y Presper Eckert, el primero director del Departamento de Física de Ursinus College, en Pasadena, y el segundo, un experto en tubos de vacío.

El primer proyecto, el ENIAC, fue un secreto militar. Se terminó en 1946, y calculó en veinte segundos la trayectoria de un proyectil (de la que antes hemos hablado, y cuyo cálculo tomaba tres días). La máquina fue enviada a Aberdeen (Maryland), al BRL. Pero ya era muy tarde, la guerra había terminado, y dejó de pensarse en ella para los cálculos de balística. Sin embargo, permaneció en funcionamiento hasta 1956, y fue muy útil en el diseño de la bomba H. Los cálculos eran larguísima, y las comprobaciones de las distintas hipótesis tomaban mucho tiempo. Gracias al ENIAC, los diseños originales sufrieron cambios y mejoras importantes. Para los norteamericanos, fue una lástima que el ENIAC no hubiera podido terminarse antes. Hubieran economizado mucho personal y tiempo. En época de guerra, el BRL había llegado a contar con un personal de hasta trescientas calculistas.

Al finalizar la guerra, Von Neumann se había trasladado de nuevo a Princeton. Estaba muy interesado en desarrollar otras máquinas, cuya configuración u organización fuera una simplificación del cerebro humano. Y con este fin, se reunía periódicamente con científicos de otras ramas de la ciencia relacionadas con su proyecto: la neurología y la psiquiatría. El Departamento de Defensa de los Estados Unidos le seguía la pista allá donde fuese, pues había sido parte fundamental en el proyecto ENIAC. Y de nuevo, por necesidades militares se le asigna otro nuevo proyecto. En este caso es la Armada la que se lo ofrece, pero luego lo utilizan las Fuerzas Aéreas y la Comisión de Energía Atómica. Al igual que en otras ocasiones, colaboran en el proyecto empresas civiles que nada tienen que ver con armamento, como la RCA, muy interesada por entonces por los tubos de vacío. El resultado es el IAS, una máquina velocísima y muy eficaz, de cuyas ventajas y problemas hemos hablado ya en otra ocasión. Este ordenador terminó pronosticando el tiempo, en un laboratorio meteorológico de la Armada. Pero diseños posteriores basados en la misma máquina han dado ordenadores tan potentes como el ILLIAC IV, fabricado para el Pentágono, y uno de los ordenadores más potentes del mundo durante muchos años.

Los sistemas basados en el conocimiento (II)



XISTEN numerosos paradigmas de representación del conocimiento. Hemos visto cómo puede utilizarse un esquema basado en las redes semánticas, o se puede usar una estructura de tripletas objeto-atributo-valor

(O-A-V), o parejas atributo-valor. En relación con ellas aparecen con alguna frecuencia los llamados *marcos de discernimiento* o, simplemente, *marcos* (en inglés "frames"). Por otro lado, se utilizan también ampliamente los formalismos de la lógica (tanto lógica de proposiciones como lógica de predicados) para representar el conocimiento. De todo ello vamos a hablar en las páginas que siguen.

Marcos ("frames")

Los marcos son estructuras de datos complejas que aportan conocimiento sobre situaciones estereotípicas. Los elementos básicos son los objetos, a los que se vinculan informaciones con ellos relacionados. En su forma más elemental el marco *representativo* de un objeto aporta un conjunto de "casilleros" o "ranuras" ("slots"), que corresponden a los atributos utilizados en las ternas objeto-atributo-valor (O-A-V). Estas ranuras se "llenan" con los valores correspondientes de los atributos. En principio, por tanto, parece que un marco no aporta demasiado sobre la posibilidad de reunir varias ternas O-A-V correspondientes a un mismo objeto. Sin embargo, el concepto de marco es más amplio y supone la fijación de

una situación estereotípica (y no sólo un cúmulo de datos sobre un objeto). Además, se acepta comúnmente que en cada ranura de un marco se pueden establecer unos valores "por defecto", de aplicación inicial en tanto no se obtengan otros valores concretos (el número de colores del ave se puede fijar "por defecto" en 1, aunque en el momento de la instanciación de un marco aparezca un ave con dos colores). También es usual incluir en alguna ranura referencias ("pointers") a otras ranuras o a otros marcos, conjuntos de reglas aplicables o incluso procedimientos. Precisamente esta flexibilidad de aportar informaciones y referencias variadas en un marco es lo que les ha hecho muy utilizados.

Nombre Ave2	
Instanciación de	Ave
Nombre	Nombre
Tamaño	Pequeño, normal, grande
Patas, número	Valor por defecto: dos
Patas, tamaño	Pequeñas, normales, grandes, muy grandes
Cuello, longitud	Número (en cm.)
Colores, número	1 a 4
Colores	Bianco, negro, verde, amarillo, rojo. Contar y comparar con colores, número.
Hábitat	Averiguar tipo de comida, tamaño y acceder a hábitat

Marco correspondiente a un tipo de ave.

En el marco presentado en la figura 1, a modo de ejemplo, aparece el nombre de referencia del marco en la parte superior. La primera ranura indica que este marco es una instanciación o especialización del marco AVE: esto permite obtener las ventajas de "herencia" en este tipo de representaciones y establecer una estructuración entre los diferentes marcos. Las ranuras que siguen presentan los elementos que componen el marco y sus posibles valores; en algunas de las ranuras se indica el tipo de información que ha de incluirse, en otras se da la gama de valores entre los que debe estar el valor concreto que "llene" la ranura; en una de las ranuras se incluye una regla que hace referencia a otra ranura del mismo marco; en la última, por fin, se alude a un procedimiento incluido en otro marco.

Como se ve, en un marco está previsto incluir informaciones descriptivas (representación declarativa de datos) e información de procedimientos (representación procedural). Este fue uno de los aciertos de Minsky cuando en 1975 propuso como esquema de representación los marcos: combinar en un solo esquema la representación de tipo procedural con la representación declarativa.

Otra de las ventajas que han hecho que la representación del conocimiento en marcos se haya extendido tanto es la facilidad de su representación en LISP. En efecto, se puede establecer fácilmente una "lista de propiedades" tal que la "propiedad" corresponda a la ranura y el "valor" corresponda al contenido de dicha ranura. Posteriormente, en las sucesivas y diversas implementaciones de este tipo de estructura se ha modificado el esquema básico según las necesidades concretas de cada caso.

Existen varios lenguajes específicos para la manipulación de los marcos: FRL ("Frame Representation Language", "Lenguaje de Representación de Marcos", Pobergs y Goldstein, 1977) parte de la organización jerárquica de los marcos, lo que comporta relaciones de clasificación y generalización; KRL ("Knowledge Representation Language", "Lenguaje de Representación del conocimiento", Bobrow y Winograd, 1977) incluye el concepto de "agenda" para el manejo de los procesos y establece cómodos mecanismos de relación de los objetos; UNITS, paquete de diseño utilizado por el sistema MOLGEN (experimentación genética), etc.

■ Guiones ("scripts")

Son estructuras genéricas de descripción de situaciones (con articulaciones internas semejantes a las de los "marcos") que representan sucesiones de situaciones tipo. Se han utilizado en el análisis de textos y tratamiento del lenguaje natural. Una "escena" está constituida por campos y contiene acciones que evalúan los valores que satisfacen cada campo. Existe una fuerte interconexión entre los campos debido a la estructura intencionalmente secuencial de la descripción. Se incluyen "planes" que describen en cada escena las opciones posibles para conseguir el objetivo deseado.

La descripción de historias mediante guiones de este tipo, como base previa al análisis de textos, ha sido muy utilizada.

■ Expresiones lógicas

La utilización de expresiones lógicas supone una aproximación al problema de la representación del conocimiento diferente, en alguna medida al menos, de las anteriormente descritas. Mediante la "lógica de proposiciones" se representan los hechos de un modo formalizado en unidades elementales (proposiciones) que sólo pueden adoptar uno de dos valores: verdadero o falso; "el avestruz tiene dos patas", es un ejemplo de proposición simple verdadera.

Las proposiciones simples se combinan entre sí mediante un conjunto de "conectivas" muy restringido; se utiliza la conjunción ("Y", su símbolo es " $\wedge$ "), disyunción ("O", símbolo " $\vee$ "), la negación ("NO", símbolo " $\neg$ ") y la implicación ("IMPLICA", símbolo " $\rightarrow$ "). De este modo formalizamos expresiones del tipo "el avestruz tiene dos patas Y es de color negro O es de color blanco" (que simbólicamente se escribirá $p \wedge (q \vee r)$, siendo $p \equiv$ "el avestruz tiene dos patas"; $q \equiv$ "el avestruz es negra", $r \equiv$ "el avestruz es blanca") o bien, "que el avestruz sea un ave IMPLICA que puede volar" ($s \rightarrow t$, con $s \equiv$ "el avestruz es un ave"; $t \equiv$ "el avestruz puede volar"). Como se ve, en el primer caso se han combinado en una proposición tres afirmaciones (proposiciones simples) que pueden corresponder a tres ranuras de un marco o a tres arcos de una red semántica, afirmando la necesidad (caso del Y) o la opcionalidad (caso del O) de que sean ciertas las proposiciones componentes para que lo sea la

proposición compuesta de todas ellas. En el segundo caso, se afirma que el hecho de que el avestruz sea un ave supone que tiene la propiedad de volar como todas las aves; se está afirmando la herencia de las propiedades del ave (una de ellas, en concreto) por parte del avestruz (instancia de la clase "ave"), en vez de establecer la relación jerárquica de pertenencia del avestruz a la clase correspondiente.

En la lógica de proposiciones se manejan las frases del lenguaje (las proposiciones) como unidades completas sin formular (formalizar) su "contenido interno": si en vez de considerar como una unidad la frase "el avestruz tiene dos patas" queremos analizar su contenido, examinaremos que en la frase aparece una "propiedad" (tiene dos patas) que se puede representar por el símbolo DP (en lógica de primer orden se llama "predicado") y que se afirma que esa propiedad la tiene el avestruz (llamémosle "a"): podemos, por tanto, escribir —como se suele hacer en matemáticas con las funciones— que DP (a) (es decir, que se cumple "DP" para el elemento "a"). Esto permite comparar lo que sucede con otro animal ("b", por ejemplo) que también tiene dos patas (será verdad, por tanto, que DP (b)), o formular más precisamente la frase anterior: en efecto, la afirmación "el avestruz tiene dos patas" normalmente (dependerá del contexto) se referirá no a un avestruz en concreto, sino a cualquier avestruz (es decir, es equivalente a "todas las avestruces tienen dos patas"), por lo que se podría formular como $\forall x \text{ DP}(x)$ ("cualquiera que sea 'x' del conjunto de las avestruces, se cumplirá que DP (x)") o, incluso, introduciendo el predicado "ser avestruz" como A (x), formalizar $\forall x(A(x) \rightarrow \text{DP}(x))$ ("para cualquier animal el ser avestruz implica tener dos patas").

Se pueden formalizar mediante predicados no sólo *propiedades* de los elementos (como en el caso de A (x) "x es avestruz" o DP (x) "x tiene dos patas"), sino *relaciones* entre varios elementos, como cuando escribimos P (x,y) "x es padre de y" o ENTRE (x, y, z) "x está situado entre y y z".

Naturalmente, cada expresión simple en lógica de primer orden (con predicados) puede ser considerada y tratada como una proposición y, por tanto, unida a otras proposiciones mediante las conectivas vistas anteriormente. Por ejemplo, la frase anterior "el avestruz (todas las avestruces) tiene dos patas Y es de color negro O de color blanco" puede formalizarse como $\forall x (A(x) \rightarrow \text{DP}(x) \wedge (B(x) \vee N(x)))$, haciendo B (x) = "x es blanca"; N (x) = "x es negra". Como se ve, aparte de los predicados que ya conocíamos, los que acabamos de definir y el "cuantificador universal" ($\forall x$) aparecen en la expresión las "conectivas" $\rightarrow$, $\wedge$, $\vee$ uniendo las distintas proposiciones componentes de la frase total.

La introducción de la lógica de primer orden (este segundo nivel de formalización que estamos comentando) con el manejo de predicados que indican propiedades o relaciones, con los símbolos de cuantificación ($\forall x$ = "para todo x" = "cualquiera que sea x" y $\exists x$ "existe x" = "hay, al menos, un x que cumple") y la utilización de variables que pueden tomar valores entre los elementos de un conjunto, se puede realizar una formalización de las frases del lenguaje natural más próxima a la realidad, pero el manejo de estas informaciones es más complejo. Existen numerosos sistemas basados en el conocimiento contruidos con bases de conocimiento formuladas en lógica proposicional, pero cada día se van extendiendo más los sistemas capaces de manejar también lógica de primer orden.

TERMINOLOGIA

Algoritmo de concordancia de «rete». Algoritmo para determinar con eficacia cuáles son las reglas que se pueden satisfacer por el contenido de la memoria de trabajo, en cada ciclo de reconocimiento, considerando las ligazones entre patrones y datos. Este algoritmo aprovecha la redundancia de los sistemas de producción, almacenando resultados parciales del cómputo de la concordancia, de forma que no necesitan volver a computarse más tarde. Fue desarrollado por Charles L. Forgy y le puso este nombre porque se utiliza en inglés una palabra antigua «rete», que significa «red» (aunque la palabra usual hoy en día sea «network»).

Arbol de comprobaciones. Estructura de datos en forma de árbol, en la que el nodo raíz representa el teorema que debe ser probado, y los nodos hijos los teoremas que una vez comprobados bastarán para probar el teorema padre. Los árboles de comprobaciones pueden ser árboles and/or.

Asociación (—binding). Asociación entre una variable y un valor para esa variable que se mantiene dentro del alcance de una regla, una llamada a la función, o la llamada a un procedimiento. Se llama también «ligadura».

BNF. Siglas de Backus-Naur. Form. (Ver **Forma Backus-Naur**.)

Cálculo de predicados. Lenguaje formal de la lógica clásica que utiliza funciones y predicados para describir las relaciones entre entidades individuales.

Ciclo. Cada paso de iteración de un bucle. En sistemas de producción, una ejecución está formada por ciclos de actos de reconocimiento reiterados. En el algoritmo de concordancia de Rete, los ciclos de concordancia tienen lugar cada vez que se añade o se elimina un elemento de la memoria de trabajo.

En sistemas de activación difundida, la activación se propaga de forma incremental durante las series de ciclos de activación.

Cláusula de Horn. En programación lógica, las cláusulas de Horn son expresiones unidas por «or» con, al menos, una proposición positiva. Así, una cláusula de Horn tiene la forma siguiente «No A o No B o ... o No C o D». La programación lógica se realiza con una mayor eficacia restringiendo el tipo de aserciones lógicas a cláusulas de Horn, del mismo modo que los sistemas de producción exigen normalmente disponer del conocimiento en forma de reglas si-entonces.

Contexto. Un estado, dentro del proceso de resolución del problema. En un sistema de producción, se puede representar el conjunto por un elemento de la memoria de trabajo específico, al que se designa con frecuencia con el nombre del elemento del contexto o elemento de control o subobjetivo. Es frecuente que las tareas aislables que deban realizar los sistemas de producción puedan partirse en subtareas que una vez iniciadas puedan ejecutarse hasta completarse.

Las reglas que constituyen esta tarea tienen cada una de ellas elementos de condición que deben concordar con el elemento de contexto asociado.

Disparar. Ejecutar el conjunto de acciones especificadas en el lado derecho de una regla o instanciación. Este término deriva de la neurofisiología, ya que se dice que las neuronas se «disparan» cuando generan una acción.

Forma Backus-Naur. Lenguaje formal para expresar gramáticas de contexto libre. Una gramática es un conjunto de reglas de sobreescritura, cada una de las cuales dispone de un lado izquierdo y un lado derecho, separados por el símbolo de metalenguaje : $\Rightarrow$. El lado izquierdo de cada regla es un símbolo no terminal de la gramática, y el lado derecho es una secuencia de símbolos no terminales y terminales. Los símbolos no terminales suelen aparecer entre los símbolos $\langle \rangle$.

Algunas versiones ampliadas de la Forma Backus-Naur incluyen símbolos de metalenguaje adicionales para indicar la repetición y la alteración. Este lenguaje lleva su nombre debido a John Backus y Peter Naur, que fueron los introductores de esta forma de descripción general del lenguaje de programación ALGOL -60.

Fuentes de conocimiento concurrentes. En un sistema experto, módulos especializados que analizan los datos independientemente, y se comunican a través de una base de datos central, llamada generalmente «pizarra».

Gramática de contexto libre. Gramática para describir un lenguaje de contexto libre. Como metalenguaje para expresar una gramática de contexto libre podemos citar la Forma Backus Naur.

Inferencia. Derivación de una proposición a partir de otras proposiciones. Se puede organizar una serie más o menos compleja de inferencias para llegar, desde unas proposiciones antecedentes dadas, hasta unas consecuencias, mediante un proceso denominado de encadenamiento hacia adelante, o de inferencia dirigida por los datos; o bien se puede comenzar a partir de la especificación de las conclusiones que se desean obtener, y retroceder intentando probar los antecedentes que las justifiquen, según un proceso que se llama de «encadenamiento hacia atrás», o de inferencia dirigida por un

modelo, o también inferencia controlada por los objetivos.

Intérprete de reglas. En los sistemas de producción, otra denominación de intérprete.

Líneas múltiples de razonamiento. En resolución de problemas, técnica en la que se desarrollan en paralelo varios sistemas independientes de resolución del problema.

Lógica. Es un sistema que prescribe reglas para manipular símbolos. Muchos sistemas lógicos tienen potencia suficiente para manipular estructuras del conocimiento que incluyen cálculo proposicional y cálculo de predicados.

Lógica difusa. Modelo desarrollado para realizar razonamientos aproximados, en el cual los valores de verdad y los cuantificadores se definen como distribuciones de probabilidades, con etiquetas lingüísticas, como, por ejemplo, verdadero, muy verdadero, no muy verdadero, bastantes, pocos y varios. Las reglas de inferencia son aproximadas en lugar de ser exactas, para mejor manipular la información que es incompleta, imprecisa o inconcreta.

Métodos basados en reglas. Métodos de programación que realizan los encadenamientos hacia adelante y hacia atrás utilizando reglas SI-ENTONCES.

Modelo de sistema de producción. Estilo de programación y resolución de problemas, caracterizado por una arquitectura de sistemas de producción. Otros modelos usuales son la programación procedural (por ejemplo, en PASCAL), programación simbólica (LISP), programación lógica (PROLOG) y programación orientada al objeto (SMALL-TALK).

Monotónico. Unidireccional. En análisis, una función monotónica es aquella que o no se incrementa (sólo se mantiene y/o decrementa), o no se decrementa (sólo crece o se mantiene). En inferencias, una lógica monotónica sólo puede añadir proposiciones a una base de conocimientos, y no puede jamás eliminarlos.

Nodo de producción. Nodo especial de la red de algoritmos de concordancia de Rete que asocia una regla con el conjunto de nodos que comprueban si las condiciones de dicha regla han sido satisfechas.

Programa de un sistema de producción. Programa de aplicación escrito en un lenguaje

TERMINOLOGIA

de sistema de producción y en el que la mayor parte del problema se lleva a cabo disparando las reglas.

Red de inferencias. Todas las posibles cadenas pueden ser generadas por las reglas, en un sistema basado en reglas.

Regla de producción. Es el tipo de regla que se utiliza en un sistema de producción. Generalmente se expresa como SI condición ENTONCES acción.

Sistema de producción. Arquitectura para la resolución de problemas, que emplea un conjunto de reglas (almacenadas en la memoria de producción), una base de datos global (almacenada en la memoria de datos) y un motor de inferencia que realiza el ciclo de reconocimiento de concordancias,

resolución de conflictos y disparo de una regla.

Valores por defecto. Los programas de ordenador suelen utilizar ciertos valores preespecificados que utilizan, si no toman otros valores alternativos. A estos valores «supuestos» se les denomina valores por defecto. Los sistemas del conocimiento suelen almacenar valores por defecto, para utilizar en lugar de hechos. Como ejemplo, citemos un programa médico que supone que un paciente ha estado expuesto a la acción de un organismo determinado, si el usuario no indica que dicha exposición queda descartada.

VOCABULARIO DE INFORMATICA

Polares, coordenadas. (Ver **Coordenadas polares**).

Polling. Término inglés que significa escrutinio, sondeo. En Informática puede referirse a una técnica para identificar qué periférico ha enviado al ordenador la señal de interrupción. También puede referirse a los «sondeos» periódicos entre terminales que utilizan una única línea de comunicación para ver cuál es el que debe obtener servicio.

Poner a cero. Se refiere a dar valores cero a las variables que son numéricas, y valor nulo a las variables de cadena. Si se trata de variables gráficas, la puesta a cero consistirá en que tomen el color de fondo, y si se trata de blanco y negro, que tomen como valor el negro.

Port. Término inglés que significa puerto. En Informática indica un canal de acceso a algún dispositivo, al que están unidos los elementos de entrada/salida. Está gobernado por el procesador central, utilizando para designarlo una dirección de la memoria que depende de la organización característica de cada ordenador. También puede designarse como puerta. (Véase **Puerta serie** y **Puerta paralelo**).

Posibilidad de traslado. Posibilidad de reubicar un programa (escrito generalmen-

te en lenguaje máquina) en otras posiciones de la memoria central. Por esta razón, las instrucciones del programa no contienen direcciones de memoria absolutas, sino definibles en el momento de la ejecución del programa.

Posición. En una cadena, cada espacio sustituible por un valor, que debe estar ocupado por un carácter o dígito binario, y que puede ser identificada por un número.

Posición de memoria. Celdilla hipotética de la memoria central (representada por su dirección), que contiene un dato, o parte de una instrucción o una instrucción completa de máquina. En Informática, los términos posición, ubicación, palabra y celda son sinónimos.

Posicional, notación. Sistema de numeración en el que los números se representan mediante un conjunto ordenado de dígitos, de tal forma que el valor por cada dígito depende de su posición, además de depender de su valor. Sinónimo de representación posicional.

Potencia. En física, número de vatios con equipo, más coloquialmente. En informática, de forma algo informal, indica la dimensión y posibilidades de un determinado equipo, más coloquialmente.

VOCABULARIO DE INFORMATICA

Power. Término inglés que significa potencia. Puede referirse a la operación aritmética de elevación a potencias, o en física o electrónica a la alimentación eléctrica.

Power failure. Término inglés para indicar la caída de tensión.

Power supply. Término inglés para designar a la «fuente de alimentación»: un dispositivo general de alimentación de un aparato eléctrico o electrónico. Normalmente, la «fuente» convierte la tensión alterna que recibe de la red en tensión continua, adecuada para la alimentación de los circuitos de los dispositivos electrónicos.

Precisión. Término que indica el grado de acercamiento de un valor al valor teórico. Las precisiones elevadas suponen márgenes de error pequeños, siendo el error la diferencia entre el valor teórico y el valor obtenido por otros medios cualesquiera. Es frecuente confundir la precisión con la aproximación, que indica, por el contrario, el grado de discriminación con la que se indica una cantidad. Por ejemplo, un número de tres dígitos discrimina entre 1.000 posibilidades.

Preset. Término inglés que indica establecer una condición inicial, como, por ejemplo, los valores de control de un bucle.

Primitiva. Instrucción básica de un lenguaje o de un sistema operativo mediante la cual se pueden formular otras instrucciones más complejas.

Printer. Impresora.

Printout. Salida de impreso.

Prioridad. Criterio de disquisición, dentro de una escala jerárquica. Puede referirse a rutinas y programas o a distintos usuarios que utilizan un mismo ordenador.

Probabilidad. Posibilidad contabilizada de que se produzca un determinado suceso, dentro de un conjunto de sucesos posibles.

Procedimiento. Secuencia sistemática de

operaciones para producir un resultado específico.

Procesador. Dispositivo diseñado para desempeñar distintas funciones sobre los datos. Los ordenadores disponen de varios procesadores que son los que realizan las distintas funciones concretas (unidad aritmético-lógica, unidad de entrada/salida, etcétera). Este término hace alusión a que los ordenadores no sólo «calculan» utilizando para ello números, sino que «procesan» todo tipo de datos.

Procesador de texto. Programa que se utiliza con el fin de obtener textos bien escritos y bien presentados, facilitando al máximo las tareas necesarias para ello. Fundamentalmente modifica, escribe, transforma textos escritos en lenguaje natural, de forma fácil y sencilla.

Proceso. Operaciones desarrolladas por uno o varios procesadores para ejecutar un programa.

Proceso predefinido. Proceso que se identifica únicamente por su nombre, y que está definido en algún otro lugar.

Producto lógico. Sinónimo de la operación lógica AND.

Programa. Serie de instrucciones propuestas en forma de lista para obtener un resultado concreto. A las distintas partes de un programa general o a los programas pequeños se les puede llamar también informalmente rutinas. Puede definirse como la elaboración de un algoritmo utilizando para ello un lenguaje de ordenador. Los programas son ejecutables inmediatamente por el ordenador si están escritos en lenguaje máquina. Si están escritos en algún lenguaje de nivel más alto, deberán ser traducidos a lenguaje máquina por un intérprete o compilador.

Programa fuente. Programa escrito en un lenguaje de alto nivel o en ensamblador, antes de su compilación o ensamblaje.

Programa objeto. Programa que se obtiene compilando o ensamblando un programa escrito en un lenguaje de alto nivel, como COBOL o FORTRAN. Es un programa ejecutable inmediatamente, ya que las instrucciones están escritas en lenguaje máquina. En ocasiones no es ejecutable inmediatamente, sino que debe ser tratado por un Linker, para unir el programa principal a las subrutinas, o por un Loader, para cargarlo en la memoria principal del ordenador.

Programa principal. Término utilizado en contraposición a Subrutina, que es un pequeño programa que depende del programa principal.

Programación. Diseño, escritura y comprobaciones de un programa.

Programación automática. Utilización de un ordenador para llevar a cabo algunas de las etapas de preparación de un programa.

Programación convexa. En investigación operativa, caso particular de programación no lineal en la que la función que se va a maximizar o minimizar y sus ligaduras son funciones adecuadamente convexas o cóncavas de las variables. En contraposición con programación dinámica, lineal, matemática o cuadrática.

Programación cuadrática. En investigación operativa, caso particular de la programación no lineal en la que la función que se va a maximizar o minimizar y las ligaduras son funciones cuadráticas de las variables controlables.

Programación dinámica. En investigación operativa, procedimiento de optimización de la solución de un problema multietapas, en el que se pueden tomar un cierto número de decisiones en cada etapa del proceso. En contraposición con programación convexa, lineal, matemática o cuadrática.

Programación estructurada. Técnica de programación que permite escribir programas fácilmente legibles, autoexplicativos, fácilmente modificables, y con la documentación adecuada. Cuando los programas se

hacían tan complejos y difíciles de comprender que los propios autores apenas si sabían lo que habían escrito, comenzó a hacerse necesario estructurarlos adecuadamente. Estos programas estructurados suelen estar formados por módulos sencillos y bien definidos. Este sistema puede utilizarse para escribir un programa en cualquier lenguaje de programación, aunque los lenguajes más modernos tienen características más adecuadas para este sistema de programación. El más adecuado es el Pascal, que prácticamente obliga a programar estructuralmente.

Programación lineal. En investigación operativa, procedimiento para localizar el máximo o mínimo de una función lineal con variables sujetas a ligaduras lineales. Este término es sinónimo de optimización lineal.

Programación matemática. En investigación operativa, procedimiento para obtener el máximo o mínimo de una función sujeta a ciertas ligaduras. En contraposición a programación convexa, dinámica, lineal y cuadrática.

Programación no lineal. En investigación operativa, procedimiento para obtener el máximo o mínimo de una función cuyas variables están sujetas a ataduras, y la función, o las ligaduras o ambas cosas no son lineales. En contraposición a programación convexa, dinámica, lineal, matemática o cuadrática.

PROM (Programmable Read Only Memory). (Término inglés que significa Memoria Programable de Sólo Lectura). Tipo de memoria ROM, cuyo contenido puede ser grabado por el usuario.

Prompt. Término inglés que significa en informática un pequeño mensaje o indicación, que utiliza el ordenador o un lenguaje para avisar de alguna situación especial. El cursor puede ser considerado como un «prompt», una interrogación, en ciertos lenguajes puede indicar que el ordenador está esperando que el usuario debe introducir datos, o incluso mensajes más largos, indicando el tipo de problema o indicación adecuada.

VOCABULARIO DE INFORMATICA

Protección. Existen muchos tipos de protección. Puede referirse a protección frente a sobrecribir sobre alguna de las memorias, cuando no debía haberse hecho. Se intenta evitar la pérdida de información debida a errores. Los métodos de protección pueden ser de dos tipos, de hardware y de software. Los métodos de hardware son muchos: muescas en discos, pasadores, o, en los discos rígidos, pulsando en un interruptor situado en la unidad de disco. Las protecciones de software se realizan introduciendo instrucciones de control en el programa, que controlen la escritura. Pueden introducirse en la creación del fichero, o mediante ciertos comandos.

Protocolo. En comunicaciones, para que la transmisión «tenga algún sentido» los dispositivos deben comunicarse entre sí según criterios normalizados generalmente por Organismos Internacionales. Al conjunto de estas normas o criterios se le denomina Protocolo.

Pseudoaleatorios, secuencia de números. Secuencia de números, obtenida por un proceso aritmético definido, y que es satisfactoriamente aleatorio para algún propósito, como por ejemplo, satisfacer una prueba determinada de números aleatorios.

Pseudocódigo. Código que requiere ser traducido antes de su ejecución.

Puerta. Parte de un circuito a través de la que se realiza la entrada o salida de los datos.

Puerto. Dispositivo de un ordenador al que están unidos los elementos de entrada y salida. El procesador central controla los puertos direccionándolos mediante una dirección de memoria o mediante un número lógico, dependiendo de la organización específica de cada ordenador.

Puerto paralelo. Dispositivo en el que los datos entran y salen en paralelo. Se transmiten simultáneamente varios bits por diversos canales (los ocho bits de un byte, por ejemplo).

Puerto serie. Dispositivo en el que los datos entran y salen secuencialmente, existiendo

un único canal por el que se transmite un solo bit cada vez.

Puesto de trabajo. Suele utilizarse también con frecuencia el término inglés workstation. Indica el lugar donde se realizan las tareas con el ordenador, o algún tipo de dispositivo periférico.

Puntero. También puede llamarse indicador. Es un valor numérico que se utiliza como dirección para un elemento. Por ejemplo, un puntero puede ser un determinado registro que contiene la información sobre el registro siguiente que se va a procesar (y que no tiene por qué ser el contiguo a él).

Puntero de una pila. Es un puntero utilizado para indicar el último elemento introducido en la pila, que será el primero en salir (sistema LIFO: last in first out; último en llegar, primero en salir).

Punto a punto. En comunicaciones, conexión directa entre dos o más puntos.

Punto de entrada. Suele usarse para indicar la dirección donde comienza una determinada rutina, o una tabla (primer elemento de la tabla).

Quadratic programming. Véase programación cuadrática.

Qualified name. Nombre cualificado. Término inglés que indica un nombre de un dato que va acompañado de especificaciones correspondientes a la clase a la que pertenece, dentro de un sistema de clasificación determinado.

Quantization. Cuantificación. Término inglés que indica una subdivisión de una gama de valores de una variable en un número finito de intervalos, no necesariamente iguales. Estos intervalos suelen indicarse mediante uno de sus valores.

Quantization error. Error de cuantificación. Error producido en la conversión de una señal analógica en digital. Como esta última tiene un número finito de valores, puede ocurrir que no se correspondan siempre con exactitud con el valor obtenido en un instante dado para la señal analógica.

Quantum. Dentro de una cuantización, un suborden.

Query. Término inglés que significa consulta, pregunta.

Queue. Término inglés que significa cola. Se refiere a un conjunto de elementos organizados en una cola en espera de ser procesados.

Quick reference. Término inglés que significa referencia rápida. Suele referirse a una tabla de consulta de instrucciones comandos, códigos, etc., o a un diccionario.

Quiescing. Proceso por el cual se detiene un sistema multiprogramado por el rechazo de nuevas tareas.

Quinario. También llamado biquinario. Notación en la cual cada dígito decimal que se va a representar se considera como suma de dos dígitos de los cuales el primero puede ser cero o uno, con un peso de cinco y el segundo puede ser 0, 1, 2, 3, ó 4, con peso de uno: de este modo, un 7 (decimal) se puede representar como 12, un 9 (decimal) como 14, y un 4 (decimal) como 04.

Quisam. Método de acceso a colas, que es secuencial, y además indexado.

Quit. Término inglés que significa abandonar. Puede referirse a abandonar una determinada tarea y volver al programa anterior, o simplemente a abandonar la tarea, el fichero, etc. Es usual que sea este término el que se utilice como nombre del comando que realiza la tarea de cerrar y abandonar una aplicación o proceso.

Qwerty, teclado. Uno de los esquemas más utilizados de disposición de las teclas de un teclado. Es el standard de las máquinas de escribir y el más extendido en los ordenadores electrónicos. Lleva este nombre como referencia a las letras que aparecen en las primeras seis teclas de la primera fila (comenzando por la izquierda).

Radian. Unidad de medida de ángulos. 360 grados equivalen a 2 π radianes. Para pa-

sar de radianes a grados basta con aplicar la fórmula siguiente:

$$\text{radianes} = \text{grados} \times (3,1416/180)$$

Radix. Palabra inglesa que designa la raíz. Ver raíz.

Raíz. En un sistema de numeración, entero por el que debe multiplicarse la cifra de un número si se desea obtener su rango. También se puede designar a la raíz como base. En informática, las raíces o bases de sistema de numeración más utilizadas son la binaria, hexadecimal, y antiguamente, la decimal.

RAM. Random Access Memory-Memoria de Acceso Aleatorio. Memoria en la que se puede acceder directamente a cada uno de sus elementos. En contraposición a memorias secuenciales, en las que para acceder a un elemento, es necesario pasar por todos los anteriores o posteriores a él. Suele llamarse de forma coloquial memoria RAM a las memorias lectura/escritura en contraposición a las memorias ROM de sólo lectura.

Random access. Término inglés para indicar un acceso aleatorio. El acceso aleatorio a un dispositivo es independiente de la posición del dato que se desea inspeccionar, y del último inspeccionado.

Recurso. Medio disponible para desarrollar alguna tarea. Puede tratarse de máquinas, herramientas de algún tipo, energía, materias primas, cualquier elemento que nos sirva para alcanzar el objetivo. El ordenador pone a nuestro alcance unos recursos, entre los que citaremos la memoria, unidad aritmética, etc. El sistema operativo es el que gestiona el uso de los recursos.

Red. Conjunto de dispositivos unidos unos a otros por algún medio. En informática se trata de una red de ordenadores o periféricos, unidos unos a otros con el fin de realizar determinadas tareas.

Redondeo. Aproximación de un número. Se redondea las cifras menos significativas eliminando la última cifra y añadiendo una uni-

VOCABULARIO DE INFORMATICA

dad a la penúltima cifra (redondeo por exceso), o bien eliminando la última cifra y dejando el resto tal cual (redondeo por defecto). No hay que confundir el redondeo con el truncamiento, que consiste en eliminar las cifras decimales que se considere, sin más.

Registro. Uno de los elementos constitutivos de un fichero. Los registros, a su vez, están divididos en campos. Puede referirse a un registro físico, tratándose en este caso de un grupo de datos, que se transfieren en bloque a la memoria de masa, (sinónimo de bloque o sector). Estos registros físicos son controlados directamente por el sistema operativo. También puede ser un registro lógico, en cuyo caso se trata de datos de un programa que están organizados formando un bloque dentro del programa. Los registros lógicos se pueden controlar mediante instrucciones.

Otra acepción de registro absolutamente distinta de las anteriores: se refiere a un circuito electrónico que sirve para almacenar de forma rápida datos de volumen muy reducido (8, 16 ó 32 bits). Los registros son una de las partes más dinámicas del ordenador. Entre ellos cabe citar al indicador de pila, registro de estado, contador de programa, etcétera.

Es importante indicar que los registros que hemos indicado primero se llaman en inglés records. El segundo tipo de registros de hardware se llaman en inglés registers.

Registro de estado. Registro de hardware situado en la Unidad Central de proceso, y cuyo cometido consiste en indicar algunas de las situaciones más importantes que se producen durante un proceso en el ordenador. Puede indicar, por ejemplo, si se produce un arrastre de una operación, si ésta es cero, si se intenta superar la capacidad máxima de memoria disponible, etcétera.

Registro índice. Registro situado en la Unidad Central de proceso, que permite calcular direcciones absolutas de la memoria. Para obtener la dirección, se toma el valor indicado en el registro y se le suma un desplazamiento relativo.

Relativa, codificación. Codificación que utiliza instrucciones máquina con direcciones relativas.

Relativa, dirección. La dirección relativa se obtiene restando de la dirección absoluta la dirección base.

Reliability. Término inglés que significa fiabilidad, es decir indica la probabilidad de que un dispositivo funcione sin fallos durante un período específico de tiempo.

Relocalizar. En programación, mover una rutina de una posición de almacenamiento, a otra, y ajustar las referencias de direccionamiento de forma que la rutina pueda ejecutarse en su nueva posición.

